OPF 问题简介以及在编程语言中的实现

陈皞翰

2024 年 11 月 18 日

摘要

本文简要概述交流电最优潮流问题的基本数学物理原理，以及在 Python,Matlab,Julia 等语言中的实现。简要介绍了 MATPOWER 工具箱，Julia 中的 PowerModel、PowerPlot 库以及 Python 中的 PyPower 库的用法。

关键词： OPF　Python　MATLAB　Julia

一、交流电网最优潮流简介

1.1 符号说明

符号	说明
$V(t)$	电压实际值
u_m	交流电的峰值电压
ω	交流电的频率
$I(t)$	电流实际值
i_m	交流电的峰值电流
ϕ	交流电电压初始相位
ϕ'	交流电电流的初始相位
U_m	交流电的有效电压
I_m	交流电的有效电流
P_m	交流电的有功功率
Q_m	交流电的无功功率
S_m	交流电的功率
Z_{mn}	两节点间的阻抗
Y_{mn}	两节点间的导纳

1.2 交流电的复数表示

1.2.1 交流电的正弦量表示

在近似于正弦波的交流电中，交流电的电压可以简单的表示为

$$V(t) = u_m \sin(\omega t + \phi) \tag{1}$$

其中 u_m 是交流电的平均电压，ω 是交流电的频率，ϕ 是交流电的初始相位。

相应的，交流电的电流可以表示为

$$I(t) = i_m \sin(\omega t + \phi') \tag{2}$$

注意，这里的 ϕ' 与 1中的 ϕ 可能并不相同，这是因为电路中可能存在无功功率的影响，这种影响将在后文中提及。

1.2.2 交流电的相量（复数）表示

由于交流电网频率恒定，故在之后的分析中我们略去 ω。在稳定的电网中，由于电流和电压的周期性，我们可以忽略时间参量，只对一个周期内的有效值进行研究。因此我们在这里引入相量图。相量图是一种只关注量的模长以及相位的图形，因此我们可以在相量图中表示交流电的幅值和相位。将相量图放入复平面中，相量的模长对

应对应复数的模，向量的相位对应复数的幅角。于是在相量图中可以将交流电表示为复数形式以及对应的极坐标形式

$$V = a + bi = U_m \angle \phi = U_m(\sin\phi + i\cos\phi) \tag{3}$$

其中电压幅值 $U_m = \sqrt{a^2 + b^2}$，电压初始相位 $\phi = \arctan(b/a)$

相应的，对于电流也有

$$I = a' + b'i = I_m \angle \phi' = I_m(\sin\phi' + i\cos\phi') \tag{4}$$

其中电流幅值 $I_m = \sqrt{a'^2 + b'^2}$，电流初始相位 $\phi' = \arctan(b'/a')$

1.3 电路基本定律

1.3.1 欧姆定律

欧姆定律给出了电路中电流和电压、电阻的关系。

$$I_{nm} = \frac{V_n - V_m}{Z_{nm}} = Y_{mn}(V_n - V_m) \tag{5}$$

其中 I_{nm} 是 n 节点与 m 节点的电流，V_n 是 n 节点的电压，V_m 是 m 节点的电压，Z_{nm} 是 n 节点到 m 节点的电阻，$Y_{mn} = \frac{1}{Z_{nm}}$ 是 n 节点到 m 节点的电导。

1.3.2 基尔霍夫定律

基尔霍夫定律表明，电路中的流入流出节点电流的代数和为 0。

$$\sum_m I_{nm} = I_n \tag{6}$$

其中 I_n 是流入 n 节点的电流。

1.4 交流电网的功率类型

1.4.1 瞬时功率

不失普遍性地，在任意电路中，电压波形和电流波形可能不是纯正的正弦或稳恒直流，这样在任意时刻的电压瞬时值乘以同一时刻的电流瞬时值就得出了一个功率瞬时值。功率瞬时值可能是正也可能是负，说明电路里的能量流时而从电源流向负载，时而又从负载返回到电源。

$$p = vi \tag{7}$$

p 是瞬时功率，v 是瞬时电压，i 是瞬时电流

1.4.2 视在功率

视在功率是指在交流电网中“看起来”的总功率，也就是的瞬时功率绝对值按时间积分后的结果。该功率可解释为电网中总的能量交换流量。于是，电网配置时，需要以视在功率为标准进行配置，而不是有功功率或无功功率。但是在 OPF 问题中，我们一般考虑有功功率和无功功率的限制。在稳定电网中，功率可以表示为电流和电压在复数域上的乘积

$$S = V \cdot \bar{I} = P + Qi \tag{8}$$

其中，$|S| = \sqrt{P^2 + Q^2}$ 是视在功率，P 是有功功率，Q 是无功功率

1.4.3 有功功率

在上述能量时而流入，时而流出的过程中，一个周期流入和流出的能量并不一定不相等，二者的差值就是负载实际消耗掉的电能，该电能除以时间周期就是有功功率。有功功率可以理解为被电网中用电器实际消耗用于做工的电功率。

1.4.4 无功功率

在电网中，存在大量容性原件和感性原件。在交流电的一个周期中，它们会先将输入的能量储存在电场或者磁场中，而后再将能量输出。事实上，（理想的）纯容性原件和纯感性原件在电路中是不消耗能量的，在一个周期中输入多少能量就输出多少能量。于是，这样额外的不对外做功的能量交换就被称作无功功率。

在纯电阻电路中，并不会产生无功功率，因为电路中的电阻在一个周期中不会储存能量，也不会输出能量，因此无功功率为零。在含容性原件和感性原件的电路中，由于电容和电感的存在，会产生感应电动势与发电机的电动势进行叠加，导致电路实际电压并不等于发电机电压，而是叠加了某一相位差。实际电流和发电机产生的电动势因此有了相位差，发电机克服相位差发电就产生了无功功率。无功功率的计算方法在前文已经给出，这里就不再赘述。

1.5 最优潮流问题简述

最优潮流（Optimal Power Flow, OPF）是电力系统中一个重要的优化问题，它旨在通过调节电力系统中可用的控制变量（例如发电机的输出功率、变压器的抽头位置等），在满足一系列运行约束条件的前提下，寻找使系统某项性能指标（如发电成本、网络损耗等）达到最优值的潮流分布。最优潮流问题的核心在于同时考虑电力系统的经济性和安全性，确保系统运行的高效性和稳定性。

最优潮流问题的数学模型由三个部分组成：目标函数，控制变量和约束条件。

1.5.1 目标函数

一般而言，最优潮流问题目标函数有以下两种：最小化发电成本和最小化网络损耗。最小化发电成本目标函数如下:

$$\min \sum_{i=1}^{n} C_i(P_i) \tag{9}$$

其中 $C_i(P_i)$ 是发电机的成本函数，P_i 是发电机的有功输出功率。一般而言，P_i 是一个非凸函数，故而 $C_i(P_i)$ 也是一个非凸函数。消耗功率的一般表达式为

$$S_n = P_n + iQ_n = V_n\bar{I_n} = V_n \sum_m (V_n Y_{nm} - V_m Y_{nm}) \tag{10}$$

P_n 即为 S_n 的实部,Q_n 即为 S_n 的虚部。易见 P_n 是一个非凸函数，从而发电功率 P_i 也是一个非凸函数。

最小化网络损耗目标函数如下:

$$\min \sum_{(k,l)\in E} Z_{kl} I_{kl}^2 \tag{11}$$

其中，E 是输电线路的集合，Z_{kl} 是输电线路的电阻，I_{kl} 是输电线路的电流。易见该目标函数也是一个非凸函数，这为优化求解带来了困难。

1.5.2 控制变量

在最优潮流问题中，一般以发电机的有功出力 P_i 为基础的控制变量，发电机无功出力 Q_i 和无功补偿 Q_c 也可能作为控制变量。

1.5.3 约束条件

约束条件包括节点功率平衡方程，输电线路潮流限制，电流限制和发电机的出力限制等。

1. 功率平衡方程

线路中的消耗功率与发电机出力功率的总和相等。

$$\sum_{i=1}^{N_g} P_i = \sum_{n=1}^{N_l} P_n + \sum_{(k,l)\in E} P_{kl,loss} \tag{12}$$

$$\sum_{i=1}^{N_g} Q_i = \sum_{n=1}^{N_l} Q_n + \sum_{(k,l)\in E} Q_{kl,loss} \tag{13}$$

$$Z_{kl} I_{kl}^2 = P_{loss} + Q_{loss} \tag{14}$$

其中 (12)、(13) 分别为有功功率平衡方程和无功功率平衡方程。P_i 和 Q_i 分别为发电机的有功出力和无功出力，$P_{kl,loss}$ 和 $Q_{kl,loss}$ 分别为输电线路的有功损耗

和无功损耗。(14) 为输电线路的功率损耗方程，$P_{loss} = \sum_{(k,l)\in E} P_{kl,loss}$，$Q_{loss} = \sum_{(k,l)\in E} Q_{kl,loss}$。

2. 输电线路潮流限制

为保障输电安全，输电线路一般有最大功率限制。

$$P_{kl,min} \leq P_{kl} \leq P_{kl,max} \tag{15}$$

$$Q_{kl,min} \leq Q_{kl} \leq Q_{kl,max} \tag{16}$$

其中 $P_{kl,min}$ 和 $P_{kl,max}$ 分别为输电线路的最小和最大有功潮流，$Q_{kl,min}$ 和 $Q_{kl,max}$ 分别为输电线路的最小和最大无功潮流。

3. 电流限制

电流限制即基尔霍夫定律，如 (6) 所示。

4. 发电机的出力限制

发电机有最大和最小输出功率限制。

$$P_{i,min} \leq P_i \leq P_{i,max} \tag{17}$$

$$Q_{i,min} \leq Q_i \leq Q_{i,max} \tag{18}$$

其中 $P_{i,min}$ 和 $P_{i,max}$ 分别为发电机的最小和最大有功出力，$Q_{i,min}$ 和 $Q_{i,max}$ 分别为发电机的最小和最大无功出力。

二、Python，Julia 和 Matlab 的对比

2.1 三种语言综合对比

1. Python：广泛用于多种领域，生态系统成熟，社区活跃，适合初学者和专业开发者。虽然性能一般，但可以通过各种库和工具提升。
2. MATLAB：专为科学计算设计，工具箱丰富，性能良好，社区活跃。但需要购买许可证，成本较高。
3. Julia：专为科学计算和高性能计算设计，性能接近 C 语言，语法简洁明了，生态系统正在快速发展。适合需要高性能和并行计算的应用。

以下表格从多个维度对比三种语言的优势和特点。

表 1 Python，Julia 和 Matlab 的综合对比

特性	Python	MATLAB	Julia
性能	一般（可通过 Cython、NumPy 等库提升）	良好（内置优化）	高（接近 C 语言）
免费和开源	是（开源，MIT 许可证）	否（商业软件，需购买许可证）	是（开源，MIT 许可证）
语法和易用性	非常友好，广泛用于多种领域	非常友好，专为科学计算设计	易于学习，简洁明了
多线程和并行计算	支持（受 GIL 限制，多进程或 Dask）	支持（需要 Parallel Computing Toolbox）	内置支持，高效
内存管理	一般（可能有较高的内存开销）	良好	高效，优化的垃圾回收机制
数值计算和科学计算	丰富（NumPy、SciPy、Pandas 等）	专为科学计算设计，工具箱丰富	专为科学计算设计，性能优越
可扩展性和互操作性	强（C、C++、Fortran、Java 等）	强（C、C++、Fortran 等）	强（C、Python、R 等）
生态系统和社区	成熟（大量第三方库，社区活跃）	成熟（成熟的工具箱，社区活跃）	发展中（活跃的社区，越来越多的库）
包管理	强（pip、conda）	强（工具箱管理）	强（Pkg 包管理器）

2.2 Matlab 求解 OPF 问题

2.2.1 常用库

Matlab 求解 OPF 问题的常用库为 MATPOWER。MATPOWER 提供了丰富且完整的电力潮流问题解决方案。其可以快速求解潮流问题，OPF 问题以及拓展的 OPF 问题。MATPOWER 开箱即用，函数简单易用，手册清晰完备。目前看来，该库仍然有一个较大的缺点是还没有完备的可视化库，可能需要自行编写可视化程序。

2.2.2 文件结构解析

在长期的研究实践中，MATPOWER 库的标准文件格式已经成为了行业共同的标准。后文 Julia 中的 PowerModels 库和 Python 中的 PyPower 库对本部分介绍的文件结构同样有很强的兼容性。接下来对文件结构做简要的介绍

表 2 参数及其对应说明

参数名	说明
文件头	
version	文件版本号，现在一般为第二版
baseMVA	基准视在功率，可用于转换标幺值

续表见后页

表 2 – 续表

参数名	说明
总线 BUS 以及总线上的负载 LOAD	
BUS_I	总线编号（从 1 开始的正整数）
BUS_TYPE	总线类型（1-> 负荷节点，2-> 发电机节点，3-> 参考节点）
PD	消耗的有功功率（MW）
QD	需求的无功功率（MVAr）
GS	分流电导（电压为 p.u. 时节点消耗的有功功率（MW））
BS	分流电纳（电压为 p.u. 时节点需要的无功功率（MVAr））
BUS_AREA	总线所属区域
VM	电压幅值（p.u.）
VA	电压相位角（Deg）
BASE_KV	电压标准值（kV）
ZONE	（损失）所属区域
VMAX	允许最大电压值（p.u.）
VMIN	允许最小电压值（p.u.）
发电机 GENERATOR	
GEN_BUS	发电机所在总线编号
PG	发电机有功功率（MW）
QG	发电机无功功率（MVAr）
QMAX	发电机最大无功功率（MVAr）
QMIN	发电机最小无功功率（MVAr）
VG	电压幅值（p.u. 仅当 opf.use_vg 非零时有效，无视总线参数）
MBASE	发电机基准视在功率，默认为 baseMVA
GEN_STATUS	发电机状态（1-> 启用，0-> 禁用）
PMAX	发电机最大有功功率（MW）
PMIN	发电机最小有功功率（MW）
PC1	PQ 能力曲线下限有功功率输出（MW）
PC2	PQ 能力曲线上限有功功率输出（MW）
QC1MIN	在 PC1 处最小无功功率输出（MVAr）
QC1MAX	在 PC1 处最大无功功率输出（MVAr）
QC2MIN	在 PC2 处最小无功功率输出（MVAr）
QC2MAX	在 PC2 处最大无功功率输出（MVAr）
RAMP_AGC	发电机有功功率爬升率（MW/min）
RANNP_10	发电机在十分钟备用模式下爬坡率（MW/10min）
RAMP_30	发电机在三十分钟备用模式下爬坡率（MW/30min）

续表见后页

表 2 – 续表

参数名	说明
RAMP_Q	发电机无功功率在 2 秒尺度下的爬升率（MVAr/min）
APF	区域参与因子
输电线路 BRANCH	
F_BUS	起始总线编号
T_BUS	终止总线编号
BR_R	输电线路电阻（p.u.）
BR_X	输电线路电抗（p.u.）
BR_B	输电线路总电纳（p.u.）
RATE_A/B/C	指定线路的潮流限制（MVA）（长时/短时/应急）
TAP	变压器匝数比
SHIFT	变压器相位偏移（Deg），正数表示相位延迟）
BR_STATUS	输电线路状态（1-> 启用，0-> 禁用）
ANGMIN	最小相位差（Deg）
ANGMAX	最大相位差（Deg）
发电机花费（目标函数）	
MODEL	模型类型（1-> 线性，2-> 多项式）
STARTUP	发电机启动成本（$）一般不计算
SHUTDOWN	发电机停机成本（$）一般不计算
NCOST	定义 n+1 项段的分段线性成本函数
COST	发电机成本函数模式 1 定义为 $\sum_{i=1}^{N} p_i k_i$
	模式 2 定义为 $\sum_{i=1}^{N} \sum_{j=1}^{n} (p_i)^n k_{ij}$

2.2.3 Matlab 求解 OPF 问题程序演示

详见附件程序

2.3 Julia 求解 OPF 问题

2.3.1 常用库

Julia 求解 OPF 问题的常用库包括:PowerModels，和其对应的可视化库 PowerPlots，求解器 Ipopt 等。事实上，以上提及的库已经能在现有理论范围内较好的对问题进行求解。PowerModels 支持求解非凸非线性交流潮流问题的复数以及极坐标表示（一般使用 Newton-Raphson 迭代法）。对于 ACPOF 的松弛形式，其同样支持 SOC（二阶锥形式松弛）以及 DC（直流松弛）的凸优化求解。值得一提的是，该库支持用

不同的支持 JuMP 生态的求解器求解，其中不乏商业求解器，能高效的给出优化问题的解。目前为止 PowerModels 还不支持分流器“shunt”以及电池“storage”。

对于可视化库 PowerPlots，它支持绘制电网图以及包含潮流的流量分布图、电压分布等。该库默认使用 Kamada Kawai 布局算法来绘制图形。在面对节点量较大的情况时，也支持选择其它的布局算法或者自定义布局来绘制图形。PowerPlots 提供了丰富的图形定制支持，可以自定义图形的颜色，大小，线条样式，图例样式等。

2.3.2 求解函数

PowerModels 支持 MATPOWER 的文件格式，我们可以统一导入 MATPOWER 数据集文件。详见附件程序

2.4 Python 求解 OPF 问题

2.4.1 常用库

Python 求解 OPF 问题常用库包括 PyPower，PandaPower 等。我们在此简要介绍 PyPower，PandaPower 是基于 PyPower 开发的库，在此不做详细介绍。PYPOWER 是一个基于 Python 的电力系统分析工具包，旨在提供一个开放源码的平台，用于电力系统分析和研究。PYPOWER 是 MATPOWER 的 Python 版本，继承了 MATPOWER 的许多功能和特点，但使用 Python 语言实现，因此可以利用 Python 生态系统中的丰富资源。但是现阶段 PYPOWER 已经不再继续更新，和更新版本的 Python 以及其它依赖库可能会有冲突。因此本文只给出一个 PYPOWER 的例子，而不再详细介绍。

2.4.2 数据结构介绍

PYPOWER 使用 Python 中的字典数据结构来表示电网数据，数据结构如下:

mpc = {’bus’:[二维 np.array，行列定义与 MATPOWER 中的 BUS 表格一致]
’branch’:[二维 np.array，行列定义与 MATPOWER 中的 BRANCH 表格一致]
’gen’:[二维 np.array，行列定义与 MATPOWER 中的 GEN 表格一致]
’gencost’:[二维 np.array，行列定义与 MATPOWER 中的 GENCOST 表格一致]
}

2.4.3 求解函数

详见附件程序

三、总结

由上述内容可见，在实际的应用过程中，我们推荐采用 Julia 中的 PowerModels 库以及 PowerPlots 库对 OPF 问题进行求解和可视化分析。这两个库有较为完善的功能，可以满足大多数的应用需求，同时也方便配置，简单易上手。对比来看，现有的 Python 库版本老旧，使用中问题较多，还需要进一步开发。而 MATPOWER 工具箱已是行业标准，可以满足大多数的应用需求，但是现有的可视化库不够成熟，需要进一步开发。Julia 和其相应的库则成熟稳定且可视化效果好，可以采用。