简介

电力系统潮流计算是电气工程本科的基础专业课，是稳态分析的主要工具之一。它的目标是在确定电力系统负荷以及发电厂有功功率出力的情况下确定各节点电压的数值，同时确定各个线路节点的功率流向以及发电厂的有功无功功率。虽然电力系统本质是是一个电路图，并且以节点电压法作为分析的基础，但是传统教科书上的电力系统潮流分析往往是从单独的节点方程来进行教学，并且也没有提供现代化的计算机编程实践。本文将以向量化的理论和目前流行Rust语言，结合现今先进的软件工程技术来介绍我开发的Rust语言编写的开源潮流计算程序：RustPower，供同志们技术交流学习以及教学参考使用。

技术理念与实现动机

目前，网络上开源的电力系统稳态分析软件有MatPower，PyPower，PandaPower和PyPSA，这些开源软件都采用了Matlab和Python环境中的数值计算以及稀疏矩阵库，对于工科的科学计算任务来说编写起来十分友好。但是Matlab和Python都是动态脚本语言，没有静态类型检查，并且性能也受到限制（Python目前还有全局锁），Matlab本身也需要商业授权才能合法使用。此外，Python动态的语言特性带来方便的同时也使得在编译时检查出程序错误变得困难。
相对而言，采用Rust语言将有以下优势：

1. Rust是静态的内存安全的语言，相对于C++而言它无须垃圾回收器也不需要用户手工管理内存分配就可以确保内存正确分配和释放，可以在实现同等性能的算法程序的情况下避免C++和C语言程序的内存泄露和野指针的问题，十分有利于复杂的并行计算程序，并且获得远超Python程序的内存占用以及运行效率的优势。并且Rust也可以用于嵌入式的环境，甚至在单片机上运行。
2. Rust 不采用继承关系，因为传统面向对象编程中的继承会导致随着软件修改而逐渐膨胀的依赖关系。例如，对基类接口的修改不可避免地会破坏所有子类的实现，这对软件升级和重构是灾难性的。Rust 采用组合的思路，通过使用 Trait 作为接口来修饰数据结构体，从而实现面向对象的多态特性。这一特点尤其适用于编写数值计算程序。
3. Rust 的生态环境和开发工具非常完善。相对于 C++ 主要依赖于标准范围外的 CMake 进行构建，Rust 将开发环境、软件包管理、模块管理、单元测试和语法命名规则等全部集成在语言本身中，无须一套自定义的复杂规范和生成工具，使得使用 Rust 的启动成本低于 C++。

另一方面的动机来源于这些年来的实践以及电力系统电磁暂态和实时仿真的研究成果，我相信可以用电路图理论和线性代数语言来描述稳态潮流计算问题并且实现更加高效简洁的程序。这一点是PyPower和Pandapower等程序并没有做到的。下面就解释我在实现这个程序中依据的理论基础。

最初的理论，最初的电路仿真

本文以及本程序实现潮流计算的基本理论自然是人尽皆知的基尔霍夫电流定律（KCL）和节点电压法，电力系统稳态，机电暂态和电磁暂态都源于此。对于一个RLC构成的线性电路，有

$$
\begin{aligned}
\sum \boldsymbol{i}_{out} &= \boldsymbol{i}_L + \boldsymbol{i}_C + \boldsymbol{i}_G \\
&= Y_L \int \boldsymbol{v} dt + Y_C \frac{d\boldsymbol{v}}{dt} + Y_G \boldsymbol{v} = \boldsymbol{s}, \\
Y_L = D_L^T [\frac{1}{\boldsymbol{L}}] D_L, Y_C &= D_C^T [\boldsymbol{C}] D_C, Y_G = D_G^T [\boldsymbol{G}] D_G,
\end{aligned} \tag{1}
$$

其中，D 是有向关联矩阵，其行对应物理组件，列对应节点；L 是电感，C 是电容，G 是电导，$\boldsymbol{s}$ 是由源引起的电流注入向量。D 是一个将全局节点电压 $\boldsymbol{v}$ 转换为端口电压的变换矩阵，而 D^T 可以将支路电流分散到节点注入向量中。Y 矩阵是不同类型组件的加权拉普拉斯矩阵，也称为导纳矩阵，在求解电网方程系统中起着重要作用。$[X]$ 表示一维向量 X 的对角化矩阵。(1)这个形式的KCL公式使得任意类型的电气元件都可以被表达为支路组件形式。对这个连续时间域的公式进行拉普拉斯变换就可以轻松得到稳态和机电暂态分析的基本公式。

$$
\boldsymbol{I}_{bus} = Y_{bus} \boldsymbol{V}_{bus} \tag{2}
$$

在这里的$\boldsymbol{I}_{bus}$是注入到节点的电源造成的节点注入电流，经过拉普拉斯变换后向量和矩阵内都为复数，也就是在频域或者相域工频的值。拉普拉斯变换导纳的细节我就不在这里赘述，这都是教科书写烂的东西。RustPower中就是依据这个公式去组装Y_{bus}的。在RustPower中，数据结构内所存储的导纳和初始组装的矩阵全部都是真值导纳，而不是每个支路归算后的导纳，这和开源实现也是一个巨大的区别，但是却可以和EMT仿真所需参数保持算法一致性，是统一模型实现三种电力系统分析的第一步。因为数据参数可能来自于多个电压等级区域，每个系统按照自己的电压等级归算万类还得按照最终计算的需要调整基值，这是给自己添麻烦的做法。

有了这个，很容易发现$S = diag[V] I^*$, 上标*代表共轭，于是得到

$$
\boldsymbol{S}_{bus} = diag[\boldsymbol{V}] (Y_{bus} \boldsymbol{V})^* \tag{3}
$$

关于用这个公式做EMT仿真的其他详情可见于我的论文
ECSGrid

T. Cheng, T. Duan and V. Dinavahi, "ECS-Grid: Data-Oriented Real-Time Simulation Platform for Cyber-Physical Power Systems," in IEEE Transactions on Industrial Informatics, vol. 19, no. 11, pp. 11128-11138, Nov. 2023, doi: 10.1109/TII.2023.3244329.

值得留意(在我看来是令人担忧)的是，在一般的公开文献中，常见的只有对单个节点的基尔霍夫电流定律（KCL）分析。教材中也通常只提供单个节点的潮流计算公式。然而，这种矩阵表达形式的理论来源于早期的电路仿真理论。上世纪70年代，计算机电路仿真的理论已经被提出并实现，早期代表性程序包括SPICE和EMTP等Fortran程序。这些早期的程序只是一些函数的集合，在现代看来显得十分原始。我国在80年代也有许多学者对这些仍处于初期阶段的EDA软件进行了研究。尽管当时的文献中也有类似的基于图论的矩阵形式表述，但并未用于实际的代码编写。

过去几十年中不使用矩阵表达形式有其深刻的原因。主要是由于过去计算机的内存和计算性能有限，而且稀疏矩阵算法直到上世纪90年代才成熟。当时的计算机连存储矩阵都十分困难，更不要说执行高性能的矩阵并行计算了。甚至连C语言编译器都没有，更不要提优化计算了。然而，现代计算机能够轻松处理这些矩阵。现在的单片机性能比过去的大型计算机更强大，编译器优化也不再需要人为地减少变量或乘法次数。对于现代家用PC的CPU和内存，即使面对几百万个节点的问题也不在话下。因此，采用矩阵形式可以更好地利用现代计算机的向量化并行计算能力。这就是为什么我强调我的程序的理论基础是基于矩阵表达的KCL公式。采用这种方式之后，RustPower中只有导纳和其拓扑这唯一一种需要组装的元器件，并且可以简单有效的处理复杂的不对称变压器相位差复数变比的情况。

目前的本科教材，无论国内外，基本都是过去的时代编写的，考试仍然要求手算。然而，手算在现代工程中已毫无意义，考试时还不是靠计算器。现代工程师应该意识到掌握计算机算法工具的重要性。或许总有人觉得，计算机和人工智能替代人的工作是令人担忧的，但是正如火车汽车跑得比人快一样，计算机比任何人都更要擅长计算。我们应该关注的是谁能更深入地理解问题，设计出优秀的程序算法和软件框架，解决实际的工程问题。如果要想建立战略层面的科学技术优势，我们必须超越现有的先进技术，让我国的工业建立在与时俱进的先进技术和理论基础之上，而不是靠比谁按计算器更快，或谁背题目更熟。

下面我们来看下用矩阵形式表达的稳态潮流计算牛顿法的雅各比矩阵是如何得到的，该方法来源于MatPower:

R. D. Zimmerman, "AC Power Flows, Generalized OPF Costs and their Derivatives using Complex Matrix Notation" MATPOWER Technical Note 2, February 2010.

我们已经知道(3)中的节点复功率，然而问题在于S_{bus}的公式出现了电压的平方因此是非线性的。 依据牛顿法我们将会得到如下迭代公式

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4}$$

这个迭代公式特指求出$f(x) = 0$这个方程的根x,然而向量和矩阵没有除法，除以$f'(x) = 0$实际转变为雅各比矩阵的逆矩阵$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}_n}^{-1}$，即

$$\begin{aligned} \boldsymbol{x}_{n+1} &= \boldsymbol{x}_n - J^{-1} f(x_n), \\ J &= \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}_n} \end{aligned} \tag{5}$$

整理成PyPower中代码就是

$$\Delta \boldsymbol{x} = -J^{-1} f(x_n) \tag{6}$$

对于潮流计算就是为了求解如下方程：

$$\boldsymbol{F}(\boldsymbol{V}) = \boldsymbol{S}_{bus} - \boldsymbol{S}_{inj} = 0 \tag{7}$$

其中 $\boldsymbol{F}$ 是构造出的在方程的根处等于0的函数，即节点功率平衡，$\boldsymbol{S}_{inj}$是电源以及负荷注入节点的功率，是一个常数。对这个复数方程可以求一阶导，得到：

$$\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{V}_m} = \mathrm{diag}[\boldsymbol{V}](Y_{\mathrm{bus}}\mathrm{diag}[\boldsymbol{V}_{\mathrm{norm}}])^* + \mathrm{diag}[\boldsymbol{I}_{\mathrm{bus}}]^*\mathrm{diag}[\boldsymbol{V}_{\mathrm{norm}}],$$

$$\frac{\partial \boldsymbol{S}}{\partial \boldsymbol{V}_a} = i \cdot \mathrm{diag}[\boldsymbol{V}](\mathrm{diag}[\boldsymbol{I}_{\mathrm{bus}}] - Y_{\mathrm{bus}}\mathrm{diag}[\boldsymbol{V}])^*,$$

$$\boldsymbol{V}_{norm} = \frac{\boldsymbol{V}}{|\boldsymbol{V}|},\ \boldsymbol{I}_{bus} = Y_{bus}\boldsymbol{V}$$

这个导数是复数形式的，还有二阶导数构成黑塞矩阵的原理在MatPower的文献中用于最优潮流计算。复功率导数取出实部虚部就能得到PQ对应的导数。
其迭代格式的雅各比矩阵（实数PQ矩阵）和电压向量为:

$$\begin{aligned} J &= \begin{bmatrix} \frac{\partial \boldsymbol{P}_{bus}}{\partial \boldsymbol{\theta}} & \frac{\partial \boldsymbol{P}_{bus}}{\partial \boldsymbol{V}_m} \\ \frac{\partial \boldsymbol{Q}_{bus}}{\partial \boldsymbol{\theta}} & \frac{\partial \boldsymbol{Q}_{bus}}{\partial \boldsymbol{V}m} \end{bmatrix}, \\ \boldsymbol{x} &= \begin{bmatrix} \boldsymbol{V}_a \\ \boldsymbol{V}_m \end{bmatrix} \end{aligned} \tag{8}$$

在实际潮流问题中，$\boldsymbol{S}_{inj}$ 的部分节点实数部分和虚数部分是提前确定的，电压$\boldsymbol{V}$ 的部分节点相角 $\boldsymbol{V}_a$ 和幅值 $\boldsymbol{V}_m$ 是提前确定的，而没有被确定的部分视为在预设条件下自动平衡，比如平衡节点的电压是确定的，但是P，Q都由最终公式得到。也就不能全部出现在最后的方程求解过程中。由此产生三种基本节点，PQ，PV，平衡节点。这些已知的量必须作为等式约束条件考虑在求解过程中，并且从雅各比矩阵中剔除相关方程。这种常量等式约束可以用如下方法化简：

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{constant} \\ \boldsymbol{x}_{unknown} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix}$$

对于这个方程组，只需要求解$\boldsymbol{x}_{unknown}$，得到

$$\boldsymbol{x}_{unknown} = A_{22}^{-1}(\boldsymbol{b}_2 - A_{21}\boldsymbol{x}_{constant})$$

注意到，在雅各比矩阵中，关于 $x_{constant}$ 等式约束项的偏导全部为0，因此 $A_{21} = 0$ ，然而该公式说明即使 $A_{21} \neq 0$ 甚至 $\boldsymbol{x}_{constant}$ 不需要为常数时我们仍然可以进行矩阵化简，相关结论在机电暂态稳定性仿真时十分有用。

矩阵切分方法

在过去的PyPower和PandaPower以及大部分我所知的开源实现中，节点的识别和雅各比矩阵剔除切分都是在迭代过程中动态选取原始系统各种节点的行列去临阵构造约束后的雅各比矩阵，在PandaPower种，他们使用了一个lookup table去加速这个过程，然而不论怎样，这样做不仅重复了无意义的内存读取和分配，而且开发工作量巨大。而在RustPower中，在进入潮流计算之前，就会产生如下的导纳矩阵，使得节点全部可以用连续内存读取 从而更容易产生合适的雅各比矩阵:

$$\begin{bmatrix} \boldsymbol{S}_{pv} \\ \boldsymbol{S}_{pq} \\ \boldsymbol{S}_{ext} \end{bmatrix} = diag \begin{bmatrix} \boldsymbol{V}_{pv} \\ \boldsymbol{V}_{pq} \\ \boldsymbol{V}_{ext} \end{bmatrix} (Y_{pmut} \begin{bmatrix} \boldsymbol{V}_{pv} \\ \boldsymbol{V}_{pq} \\ \boldsymbol{V}_{ext} \end{bmatrix})^* \tag{9}$$

这样在雅各比矩阵切片时，只需要知道有多少PV,PQ,和平衡节点，就可以切出连续的分块矩阵。而求解状态更新也不再需要匹配节点索引，可以直接向量相加。并且这个方法对于任意多的节点(0个PV节点或者多个平衡节点)以及特殊类型都可以适用，实现起来非常简单。
得到这个方法的核心在于如下数学原理：
首先让我们记调整顺序之前的电气量为S,V,Y等，调整后的为$\boldsymbol{S}^p$,$\boldsymbol{V}^p$,Y^p，通过基本的线性代数理论,我们发现V^p可以通过如下线性变换得到

$$\boldsymbol{S}^p = T\boldsymbol{S}, \boldsymbol{V}^p = T\boldsymbol{V}$$

其中，T为每行只有1个为1的元素的置换矩阵，左乘可以置换矩阵的行，右乘置换矩阵的列，它的具体内容不在这里赘述。于是可以发现

$$\begin{aligned} S^p = T\boldsymbol{S}_{bus} = Tdiag[\boldsymbol{V}]Y_{bus}^*\boldsymbol{V}^* \\ T\boldsymbol{S}_{bus} = Tdiag[\boldsymbol{V}]T^{-1}TY_{bus}^*\boldsymbol{T}^{-1}TV^* \end{aligned}$$

并且我们不难发现

$$Tdiag[\boldsymbol{V}]T^{-1} = diag[T\boldsymbol{V}] = diag[V^p]$$

进行代换就可以得到

$$
\begin{aligned}
\boldsymbol{S}_{bus}^{p} &= diag[\boldsymbol{V}^{p}](Y_{bus}^{p}\boldsymbol{V}^{p})^{*}, \\
Y_{p} &= TY_{bus}T^{-1}
\end{aligned}
\tag{10}
$$

只要将这个节点顺序变换到导纳矩阵上，就可以一劳永逸的解决矩阵切分问题。并且这个置换矩阵采用稀疏矩阵技术非常容易构造和计算，可以实现任意节点排序和顺序还原。接下来的一切都十分直接了当，直接按照迭代公式算，算完收敛了通过逆变换得到原来节点标号的计算结果。

这项本该十分稀松平常的技术是在我研究机电暂态稳定性仿真时发现并得到实验验证的，并且也用来解决电磁暂态仿真的一些建模问题，现在只是将它介绍到稳态潮流计算。我个人认为，主要在于很少有计算机算法的研究者同时对电力系统的问题有特别的需求从而产生较为深刻的认识，往往每个人只专精于自己的一小块天地，而很少出现喜欢计算机科技却学了电气工程然后头铁从变电站工程师做到学术研发电磁暂态仿真同时漂泊海外的技术激进主义博士生去思考这些问题。抛开个人经验的特殊性，我认为这证明了团结合作和知识共享的重要性，不能更新的知识和封闭的体系最终要被开放全面的体系击败。也正因为网络开放的资源和知识，OpenAI的ChatGPT所知道的知识广度才能超越任意单个人类。

变压器的变比的讨论

几乎国内看到的所有教材和公开网络文献都表示，变压器变比就是个k是实数，三相变压器变成PI导纳，然后组装到导纳矩阵永远是对称的矩阵。然而开源的MatPower等所有软件都是考虑了变压器连接组别导致的相位移动的。这样变比会变复数导致矩阵不对称。下面用变压器理论公式来展示为什么会出现这种情况:
首先把所有电气量归算到一次侧，得到

$$
\begin{aligned}
(T^{*})^{-1}\boldsymbol{i} &= YT\boldsymbol{v}, \\
T &= \begin{bmatrix} 1 & 0 \\ 0 & k \end{bmatrix}
\end{aligned}
\tag{11}
$$

其中k为变比, T可以把电流电压真值变换到高压侧。这时就已经足以说明问题，假定$k = k_m e^{i\theta}$，它的逆矩阵是

$$
T^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{k} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{k_m}e^{-i\theta} \end{bmatrix}
\tag{12}
$$

而共轭逆矩阵是

$$
(T^{*})^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{k_m}e^{i\theta} \end{bmatrix}
\tag{13}
$$

所以得到的Y是
$$
\begin{equation}
T^{*}Y T = \begin{bmatrix}y *k_{m}^{2} & y k_{m} e^{ i \theta}\\ y k_{m} e^{- i \theta} & y
\end{bmatrix}
\end{equation}
$$

只有这样才符合对电流电压进行同样的相位偏移，也就是变压器连接组别会造成的实际情况，否则就会变成一般导纳那样电流电压

$$
\begin{equation}
T^{*}Y T = T^{*}{}^{D}{T}Y_b D T
\end{equation}
$$

$其中，Y_b是支路的**真值导纳**，不要归算标幺值也不要算变比，这里变比变换T$可以乘到关联矩阵中去也可以分开，因此相关矩阵很容易构造并且计算。所以，即使是RustPower内存储的导纳全部采用真值导纳，处理这样的变压器也不会遇到困难。但是由于本身潮流计算相位就是虚拟的而且变压器的相位差在IEEE测试系统上设置是0，RustPower暂时没有实现相关功能而仍然采用常规的PI型导纳。有关这个问题欢迎各位同志和专家讨论以给出权威性的结论或测试算例。

结束语

RustPower就是基于两个非常简单的矩阵:关联矩阵，置换矩阵，高效敏捷的完成了潮流计算算法的开发且提高了性能。当然他还有许多工程上的巧妙设计，比如通过Rust的feature实现条件编译选择最合适的求解器，以及对c语言klu的自动封装，数组构成结构体的数据类型设计以及一些基本稀疏矩阵算法的实现。之后如果需要并行计算，之前的这些原理和工程设计会有很多益处。开发中ChatGPT在编写代码的过程中提供了极大的帮助（清理代码和加注释以及告诉我稀疏矩阵的一些细节），使得开发过程变得更加高效。我希望通过这个介绍，能让同志们换一种角度看待似乎以及习以为常的电力系统经典问题，并且从中获得一些启发。我相信先进的理论和技术不是靠个别人只靠存在自己脑内的天才去发展的，相反，RustPower应该能够让同志们明白，进步和创新靠的是学术知识的共享以及其核心数学理论，软件开发，工程管理和测试经验三个方面的紧密结合。我欢迎同志们为RustPower贡献代码或者提出建议。我们的国家从来都不缺少绝顶聪明的人，然而我时常觉得可惜，有这么多的仁人志士，却因为现实原因很少能够团结在一起去做些什么，而西方国家的各种开源软件和社区却像雨后春笋一样萌发，有力的推动了他们的和我们的技术革新。事实上只要我们能够将我们国家的学术理论，工程技术，和科技管理三个方面的工作者紧密的团结在一起为了共同的目标奋斗，那么一定可以攻克任何科学与技术的难关。

写作时间：2024年5月