船舶水动力数值模拟方法的一些综述介绍，第一篇：计算流体力学的一些基础知识
（黄军涛科学网博文，链接地址：http://blog.sciencenet.cn/blog-3472103-1280473.html）

船舶水动力数值模拟方法不断发展，水动力软件的也日新月异，更新很快，有很多专题的文章发表，但全面性的综述还需要更多的介绍。本人参考了一些资料，再结合自己多年的经验想做这方面的尝试，尽量多做全面性的综述介绍，但精力毕竟有限，广度上很难真真做到全面（比如没涉及网格产生的技术），深度上也很多停留在基本知识的介绍，只是希望这些工作能给一些人带来帮助，为他（她）们的科研和设计工作节省些时间，也希望抛砖引玉，见到更多的这方面的讨论和文章。初步设想通过两篇博文来完成，第一篇：计算流体力学的一些基础知识的综述介绍（只介绍和船舶水动力数值模拟相关的一些内容），第二篇：方法综述并以 CFD-OHMUGA 为例子对方法和软件功能进行更为具体的介绍。

需要指出的是本文涉及到的某些基本概念的定义直接引用了一些教材或其他资料的说法，因时间仓促和博文形式，在这里没有明确指明出处。如果需要，会做这方面的工作。

计算流体力学（Computational Fluid Dynamics: CFD）是流体力学的分支学科，是一个介于数学、流体力学和计算机之间的交叉学科，主要研究内容是通过计算机和数值方法来求解流体力学的控制方程，对流体力学问题进行数值模拟和分析。

通常的 CFD 包括 3 部分：（1）前处理（生成网格，输入计算条件），（2）求解器，（3）后处理（计算结果数据处理）。其中求解器部分是核心，主要包括并行计算要求的 DD（Domain Decomposition）方法的选用和开发，网格的分割和并行化处理，并行方法的选用，数据压缩存储和高效交换的结构设计，精确数学模型的选用，稳定精确的数值方法的选择和开发，以及并行化的快速稳定和精确的线性求解器的选用和开发，等等。
下面对一些基础的问题稍微展开来进行解释。

1.数学模型

数学模型主要是 N-S 方程，湍流模型，以及其他物理现象涉及到的数学方程。这里需要指出的是，精确模拟湍流总是一个挑战。湍流长度和时间尺度广阔的分布范围和湍流现象的复杂性，使得要解析出所有湍流尺度问题（DNS 方法）在计算上显得非常昂贵。往往是越多的尺度被解析（越少尺度被模拟），计算越耗时（也需要更好的计算机），但精度会越高。RANS(对所有尺寸进行了模型化), LES（大尺度涡直接可以解析出来，只对网格尺寸以下的尺度进行模型化），DNS（没有模型化，直接全部解析）方法是计算越来越耗时，但精度越来越高的趋势。在这种情况下，人们可以准对不同模拟问题的具体要求在计算耗时和模型的精度方面权衡考虑并作出选择。很多工程问题采用 RANS 模型基本可以取得好的计算结果。如果想对涡结构有更好的解析，一般采用 LES 模型，但 LES 在近固体壁面处还是很耗时，RANS 和 LES 优点互补的混合模型 DES 模型则是一个好的选择，在近壁面处采用 RANS，远处的流体分离区则采用 LES 模型。DNS 方法在工程现实模拟中则一般不会考虑和应用。

另外还可以采用壁面函数的方法，在近壁区采用半经验公式，并使得第一个网格点安排在边界层的粘性底层外，离壁面更远的对数层。好处： 1）对高 Reynolds 数的一些湍流模型，在边界层近壁处的低 Reynolds 数区域精度上得到一定的改善，2）一定程度上可以减少边界层中的网格数，节省计算资源，另外邻近壁面的第一个网格点到壁面的距离数值不至于太小（比如无量纲值 1.0E-7，可能会小于网格软件中判别两几何点是否重合的公差），并可以减小近壁处的网格比，这对比如特别高 Reynolds 数的原船问题的模拟计算很有帮助。

2.数值方法

2.1 数值方法的一些基本问题

1)相容性(Consistence)。

相容性是指当步长趋近于零时，离散格式的截断误差趋向于零，或者说离散格式（离散方程）趋近于微分方程。

2)稳定性(Stability)。

稳定性是指在数值求解过程中，引入的误差不会产生实质性的增长，不会导致离散方程解的失真。比如迭代求解不会发散，对于非稳态问题，解是有界的。主要用 Fourier 方法（或称 Von Neumann 方法）进行离散方程的稳定性分析。

3)收敛性(Convergence)。

收敛性是指当网格步长趋近于零的时候，离散方程的数值解趋近于微分方程的真实解。

4)守恒性(Conservation)。

流体力学的基本方程都是描述流体运动中物理量守恒律的数学方程组，采用守恒型离散格式有利于数值求解过程中保持物理量的守恒特性。在微分方程中如果所有导数系数仅为自变量的函数则称方程为弱守恒型，如果所有导数系数仅为常数则称方程为强守恒型。比如散度型方程就是守恒型方程。如果离散格式是相容的，且在求解区域内在任意网格点数和网格尺度都精确的满足离散型散度形式，则称离散方程是守恒的。守恒型的离散格式，必须是在局部和全局都满足守恒定律，比如从守恒型积分型方程为出发点的有限体积法是守恒型的离散方法。值得指出的是在大多数情况下，由于不守恒而导致的误差仅在相对粗糙的网格上才可观，但很难定量估计。

5)有界性(Boundedness)。

有界性是指数值解应该在适当的范围内（例如湍流的 TKE 为非负，浓度在 0 和 1 之间），但实际中有时很难保证，尤其是对于高阶精度的离散格式。

6)现实性(Realizability)。

由于太复杂而无法直接处理的现象比如湍流，在模型的设计应该保证物理上有现实的解。

7)精度(Accuracy)。

数值解的精度受到各种误差的影响，主要有数学模型的误差，离散方程的离散误差，计算的舍入误差。 要注意根据实际问题选择不同的湍流模型。对于离散方程的离散误差，目前在工程应用上大部分问题，一般在时间和空间上保持二阶精度，就可以得到可接受的结算结果。另外在计算语言上要尽量采用高精度的数据储存方式（比如双精度数据类型）以减少舍入误差。

2.2 偏微分方程的一些数值解法

1)有限差分法（FDM: Finite Difference Method）

有限差分法（FDM）是一类用有限差分逼近微分方程的数值方法。它的基本思想是将定义域进行网格剖分，然后在网格点上，将微商换成差商，从而把原问题离散化为差分格式，进而求出数值解。这种方法经常会和结构网格的方法相结合（比如 CFDSHIP-Iowa V4.5），可以方便的对流项的离散采用高精度的形式。

2)有限体积法（FVM: Finite Volume Method）

有限体积法以积分形式的守恒方程为出发点而不是微分方程，着重从物理观点来构造离散方程 ,每一个离散方程都是有限大小体积上某种物理量守恒的表示式 ,保证离散方程具有守恒特性。有限体积法很容易在非结构化网格中应用（比如 CFD-OHMUGA）。 目前这种方法被很多计算流体力学的商业软件大量的采用。

3)有限元法(FEM: Finite Element Method)

有限元法将求解区域细分为更小的数量有限的单元，通过变分法或加权余量法将微分方程离散。采用分段逼近的方法，先利用单元内节点构造的插值函数对每一单元进行离散处理，再组装全局的方程，然后对所有节点变量的线性方程组求解。由于传统的有限元法往往对各变量采用相同的插值方法，而流体的对流稳定格式则需要上风格式，需要特殊处理，这可能是很多商业软件没有采用有限元法来计算流体力学的原因之一。目前有很多改进的有限元方法能够处理这种对流的上风格式。比如 Streamline-upwind/Petrov-Galerkin 方法。

4)边界元法(BEM: Boundary Element Method)

边界元法将微分方程的边值问题化为边界的积分问题，并吸收了有限元的一些离散技术。与有限元需要处理整个体积区域不同，边界元只要处理边界处的曲面即可，问题降低了一维，使得单元数减少，计算速度加快，比如对求解势流问题有一定的优势。但边界元法往往适用于简单的方程，边界元法中涉及的奇异积分需要特别处理，而且产生的方程系数是满矩阵（有限差分，有限体积，和有限元都是稀疏矩阵），计算速度也受到一定的拖累。

5)谱方法 (SM: Spectral Methods)

谱方法是把解近似地展开成光滑函数(一般是正交多项式)的有限级数展开式，再根据此展开式和原方程，求出展开式系数的方程组。与有限元的分片局部近似不同，谱方法的这种解的近似是对整个计算区域的近似。此方法精度高，收敛速度快。谱方法不宜求解复杂计算区域和边界条件的问题，对于强梯度或间断问题（比如激波），误差则较大。

6)格子玻尔兹曼方法（LBM: Lattice Boltzmann Method）

格子玻尔兹曼方法是一种基于介观（mesoscopic）模拟尺度的方法。该方法相比于其他传统 CFD 计算方法，具有介于微观分子动力学模型和宏观连续模型的介观模型特点，因此具备流体相互作用描述简单、易于复杂边界条件，易于处理多相流和多孔介质等问题，易于并行计算、程序易于实施等优势。但需要分配很多内存来储存分布函数，对计算稳态问题效率不高，处理高马赫数可压缩流有困难。

7)无网格方法（MFM: Meshfree Methods）

无网格方法是在数值计算中不需要生成网格，而是按照一些任意分布的坐标点构造插值函数离散控制方程，可以方便地模拟各种复杂形状的流场和大变形的问题。但无网格方法需要继续为减少计算量，提高计算速度而努力。

2.3 网格类型

除了无网格方法（Meshfree Methods）以外， 绝大部分网格可以按照结构网格和非结构网格分成两大类。

1) 结构网格(Structured Grid)

这是一种六面体网格，最简单的是直角坐标坐标下的长方体网格，尤其是正方体网格在所有网格中质量是最好的。这种直角坐标网格的单独使用往往受限于简单的形状，但混合类型网格的焦点区域局部使用，或 Overset 网格的局部改善或加密对计算都很有帮助。曲线坐标网格尤其是非正交曲线网格可以对复杂的物体曲面采用适体网格，往往只对边界层的垂直方向设置足够多的网格，而平行方向可以适当的减少网格数量，从而既保持足够的计算精度又减少了计算工作量。船舶的边界层厚度相对船长非常小，这种适体网格的设置对船舶水动力的计算是非常有帮助的。这种网格的节点和单元很容易在程序语言中以数组的(i,j,k)进行标注，所用内存少，而且很容易对对流项采用高精度（可以是隐格式）及 TVD 格式进行数值离散（3 阶精度以上），而在离散的模板上又容易保持精致的性质（比如对于直角坐标网格而言，线性方程矩阵保持比如 5 对角矩阵等带宽小精致的特性）。这种设置往往很适合用比如 ADI(Alternate Direction Implicit)求解器对线性方程组(比如对流占优的抛物型方程)快速求

解。即便是非正交曲线网格，和非结构网格相比，压力方程的矩阵也可以保持相对精致的形式。

注意很多时候结构网格以多块网格（Multi-Block，有多个子网格组成）的形式使用，可以处理复杂计算区域和几何形状问题而又在每块子网格保持结构网格的数据结构。

2) 非结构网格(Unstructured Grid)

非结构网格可以由更一般的适合复杂边界形状的网格单元组成，比如六面体，四面体，三棱镜（五面体），金字塔等形状（五面体），更任意的多面体等（可以全部是单一类型的体单元组成，也可以是不同类型单元的各种混合），这使得它能处理比较复杂几何形状的问题，在现实的工程问题的计算中得到大量的应用。很明显结构网格本质上是非结构网格的一种特殊情况而已。非结构网格会采用和传统的结构网格不一样的数值处理方法（除非为了获得结构网格的优点，有的软件会准对六面体网格进行一些特殊处理）。有限体积法则在非结构网格的软件中被广泛的应用。和结构网格相比，非结构网格的主要困难包括：（1）网格单元的体，曲面，边和节点的排序，以及在局部和全局的本身或和邻居的对应关系的布置，以及快速搜索会更困难，尤其是在并行计算的情况下，（2）需要设计合理的数据储存结构，以便节省内存和进行数据的快速交换，（3）无论在精度和效率上，几何参数的计算会更复杂，要求更高，（4）在离散格式中所用的模板节点的分布范围更大（导致线性方程矩阵带宽比较大，分布不规律），尤其是高精度的对流隐格式需要很多节点数量的模板（使用大量的插值），这无论是对内存还是线性方程组的求解器提出更高的要求。总之，和结构网格相比，非结构网格技术更能处理复杂的边界形状，但无论是计算精度还是计算速度将面临更大的困难和挑战。

无论是结构网格还是非结构网格，从网格的均匀程度来分，可以分成：（1）均匀网格（Uniform Grid），（2）非均匀网格(Non-Uniform Grid)。非均匀网格的目的是为了在尽量保持计算精度的情况下，在非焦点区采用相对粗的网格，通过节省网格节点的数量，减少计算工作量，加快计算速度。

有时候为了适应特殊的情况，比如运动的边界或物质界面，有相对运动的多个物体等，从网格的是否运动的情况来看，可以分成（1）欧拉网格（Euler Grid 空间上固定，这种方法在 CFD 中很普遍），（2）拉格朗日网格（Lagrangian Grid, 随流体运动,完全的 Lagrangian Grid 方法在固体结构的模拟中较常见，在船舶 CFD 流场计算中没见到过），（3）混合网格（任意运动，比如 ALE 法（Arbitrary Lagrangian Eulerian Method），这种方法在船舶 CFD 计算中经常见到）。

除了需要处理物体或界面的运动或相对运动问题，有时候还要处理比较尖锐的形状，流场中梯度较大的区域等焦点区的网格的动态加密等，出现了自适应网格（Adaptive Grid）技术。自适应网格划分技术一般可分为两大类：（1）自适应网格重分布，（2）自适应网格细化。自适应网格重分布技术不断地重新定位固定数量的单元，以提高流体流动域特定位置的分辨率。自适应网格细化技术根据需要添加新单元，并删除不再需要的其他单元。比如 CFDSHIP-Iowa Version 3.03就是采用了自适应网格重分布技术，网格随着水自由面的变化也进行变形调整，（Paterson EG，2003, General-purpose parallel unsteady RANS ship hydrodynamics code: CFDShip-Iowa）。当然在边界静止和网格数量不变的情况下，也可以利用自适应网格技术对计算区域内部的网格进行重新分布，局部加密或改善网格质量。自适应网格也可以对焦点区域（比如气泡变形区域，波的破碎区域，流体边界层等）进行动态加密细化（增加数量），而在非焦点区域则动态变疏（删减数量），比如AMR技术（Adaptive Mesh Refinement, 源自 Berger MJ, 1989, Local adaptive mesh refinement for shock hydrodynamics），被广泛的应用与流体自由界面运动的界面局部精整加密。还有很多自适应网格的技术，不再一一介绍。

上面提到的自适应网格的方法，虽然可以通过网格的移动和扭曲变形来处理一些动边界

和物体相对运动的问题，但这种扭曲是有限度的，如果扭曲变形过大，网格质量变差，使的计算精度降低，严重的扭曲（比如旋转运动）则使得网格被破坏失真，导致计算崩溃。另外需要处理比较复杂的几何形状和网格的局部加密。

为了解决以上问题，引入了重叠网格（Overset Grid）技术。重叠网格技术是指通过把各自独立的网格剪裁掉重叠的多余部分后拼成一个整块网格，并通过剪裁后的各网格相互嵌套交接的边界交换数据，从而完成对整个计算区域的数值求解。重叠网格中的网格点可以分成三种类型：（1）控制方程的计算点（Active Point, 这些点是需要通过控制方程离散求解和直接输出的），（2) 交接边界插值点（Fringe Point，通过与之嵌套的其他网格的有效Donor cell插值计算），（3)洞点（Hole Point，理论上一般是不需要考虑的）。当然，在Fringe Point中还可能有一种没有对应的Donor Cell的孤点（Orphan Point），如果不处理会使计算中断，要特殊处理，尽量避免的。重叠网格三种类型的点以及插值关系的信息叫做DCI (Domain Connectivity Information）。重叠网格技术可以分成两大类型，（1）静态重叠网格（网格静止），（2）动态重叠网格（允许部分或全部网格有各自不同的运动）。

重叠网格技术方法是一般先布置一个背景网格（一般是直角坐标的网格），再加入带目标体的网格或其他各种级别的加密网格。重叠网格可以使得复杂的几何体周围的网格质量不会太差，比如对舭龙骨单独构造一个网格，再贴附和叠加在光滑船体的网格中，可以使得整体的网格质量不至于太差。在流场梯度大的地方，或流体界面处，布置加密的独立网格，可以在焦点区域得到足够高的计算精度。由于各个网格可以有独立的运动，所以可以在网格不变形（保持网格质量）的情况下很容易的计算有物体相对运动的问题（船舶问题中，一般采用ALE 法计算流体力学控制方程）。

需要补充说明的是，实际上某些时候洞点必须赋值,比如有人提到物理解有时候在不同时间层出现跳跃变化或计算发散，就是因为求时间导数时可能用到了非物理的值，该计算点在前面的时间层可能是洞点。 另外，组成Donor cell的所有点必须是Active point（大部分时候Donor cell是原始Cell体的整体，偶尔也可能是原始Cell体的某一部分但至少是四面体）。还有，如果计算区域的物理边界面处如果有网格重叠，则必须特殊处理，对重叠的面进行相互切割形成不重叠的一个面，以便能在该边界处对力，力矩，通量等物理量有精确的数值积分值。对于重叠网格来说，很大的一个挑战是处理效率或者说是速度。困难在于一些看起来是需要串行处理的办法，怎么让它并行化。还有从方法上如何让用户操作简单，也就是少人为干预，多自动处理（比如对网格可以不强设等级等）。还有如何减少Orphan point， 提高插值精度，总是一个不断提高的过程。这些方面的内容在其他地方（比如Overset-OHMUGA用户手册，或本作者公开发表的文章中有详细的描述）。

2.4 N-S 方程时间和空间上的离散方法

上面提到的有限差分法，有限体积法，有限元法等方法在时间和空间上有各自的数值离散方法。这里要介绍一些基本的概念（主要是准对有限差分法和有限体积法）。比如显格式和隐格式，稳定的对流离散格式，高精度方法，高分辨率方法等等。

1) 显格式和隐格式

显式方法从系统前面时间层的状态直接计算当前的状态，而隐式方法则通过一个既包含系统前时间状态又包含当前状态的方程来求解。很显然，显式方法比较直接简单，也不需要涉及到线性系统的求解，并行化简单且效率高。但显格式（比如前向 Euler 法）的对数值稳定的条件要求更加苛刻（比如苛刻的 CFL 条件），往往时间步长不能太大。隐格式（比如后向 Euler 法）是很常用的方法，即便采用相对大的时间步长往往还可以保持格式的稳定，但需要解线性系统。当然还有显格式和隐格式的混合格式（比如 Crank Nicolson method，时间上具有二阶精度）。对于大部分的问题，时间上离散的二阶精度一般是足够的。

2) 扩散和对流项的离散方法

对于 N-S 方程的扩散项的处理，有限差分方法中一般采用二阶精度的中心差分方法。有限体积则采用控制容积面上积分的方法。

对于 N-S 方程的对流项的处理，主要是要构造稳定的格式，于是出现了各种形式的上风格式，最开始是一阶精度的上风格式（最稳定，但精度最低）。工程中一般是需要根据实际情况选择一些不同的不低于二阶精度的格式（一般情况下二阶精度足够了），比如二阶精度的线性上风格式（ LUDS：Linear upwind difference scheme，其中 biased scheme 用的很多），QUICK 格式（三阶精度），等等。

3) 高精度方法(High Order Method)

无论在时间上还是空间上的离格式，精度一般在二阶以上的格式，叫做高精度格式。高精度格式多用于求解非稳态的多尺度的流动问题，比如研究流动的细微结构，很多高精度格式都用在湍流的模拟上，比如对于 DES 湍流模型，一般需要至少三阶精度（有的时候五阶精度）的空间的离散格式才有意义。 注意高精度往往对数值稳定性提出更高的要求。

4) 高分辨率方法(High Resolution Method)

数值解的高分辨率（注意这里高分辨率和上面提到的高精度不是一个概念）是指精确的描述对感兴趣的流动特征，精确刻画数值图形的锐利和逼真。最开始用于对-双曲方程的数值求解中为了精确地模拟激波，后来广泛的应用于计算流体力学的各种领域。典型是（1）总变差不增的 TVD（Total Variation Diminishing）格式，这种方法有利于保持高分辨率，但往往很难构造出高阶精度的格式， （2） 无振荡格式(Non-Oscillatory),比如基本无振荡格式 ENO(Essentially Non-Oscillatory), WENO（加权的 ENO: Weighted ENO），这种无振荡格式往往可以构造出很高阶精度的格式，三阶精度，五阶精度等。需要指出的是通过限制函数或限制器可以调节数值耗散和色散，是构造无振荡格式，特别是保持 TVD 的重要手段。比较典型的一些限制函数有比如：Roe's minmod (1970)，Roe's superbee (1985)，Van Albada (1977)，Van Leer (1974)等等。

2.5 不可压 N-S 方程的解法

对不可压 N-S 方程的解法，可以分成非原始变量法(Non-primitive Variables Formulations)和原始变量法（Primitive variable Formulations）。非原始变量法有比如势函数涡函数法（stream function-vorticity formulation），解三维问题效果不好，一般不太常用。原始变量法则较常用，介绍如下：

1)分离的方法(Segregated Method)

(1)投影（Projection）

投影法的依据是 Hodge 分解法则，即域内的任一矢量都可唯一地分解为一个散度为零的矢量和一个标量函数的梯度。

a) 分数步（fractional steps）

这种方法不一定是隐格式，对动量方程分步求解，并利用 Poisson 方程求解压力，并对速度进行修正，以满足质量守恒方程。

b) 一般的投影（Normal projection）

一般采用隐格式，把动量方程等式左边写成有关速度导数项和压力梯度（或除以密度）项的加和。对动量方程求散度，满足质量守恒方程后，就可以得到压力方程。求出压力后，再利用投影矢量表达式可以对速度进行修正，以满足质量守恒方程。投影方法只要求求一次压力方程和进行一次速度修正，速度较 PISO 快。隐格式则可以采用较大的时间步长。

(2) SIMPLE 系列

a)SIMPLE

SIMPLE 法(Semi-Implicit Method for Pressure Linked Equation 即：压力连接方程半隐式方法)。SIMPLE 法的步骤大致是先求动量方程可得速度的值，再求压力修正方程，求得修正后的压力，及修正后的速度值，每一时间层都是内迭代和外迭代循环多次直至收敛。

b)SIMPLEC

SIMPLEC 法(SIMPLE-Consistent，即：协调一致的压力连接方程半隐式方法)。SIMPLEC 和 SIMPLE 想类似，但 SIMPLEC 法速度修正方程更多的考虑了邻居点的影响，使得速度和压力的修正更为协调，从而使收敛更快,松弛因子也可取得更大。

c) SIMPLER

SIMPLER 法（SIMPLE Revised）对 SIMPLE 法有所改变，直接解出压力场（要求严格收敛），但保留了压力修正方程（还要再解一次压力修正方程）来修正速度，总体计算时间 SIMPLER 常比 SIMPLE 少。

d) PISO

PISO 算法（Pressure-Implicit with Splitting of Operators）是 Issa 于 1985 年提出的一种预测和多步校正方法去计算动量方程并满足质量守恒。 预测步是从假定一个压力（一般是最近历史的压力值）的动量方程中算出流体速度，校正步是满足质量守恒算出压力值，再对速度进行校正，这样的校正步骤一般需要 1-6 次迭代。相对于投影方法的一次迭代，PISO 算法耗时要长很多，但质量守恒方程的残差要小些。

(3)交错网格或同位网格的安排

交错网格的方法是把压力和速度网格点放在交错的网格位置上，可以避免压力的振荡，但这样以来对几何的计算造成额外的负担，对复杂形状的问题，比如曲线坐标的结构网格或非结构网格的问题的数值求解造成比较大的困难。

很多商业软件都采用同位网格的方法，也就是说所有的变量都放在同一套网格上。采用 Rhie-Chow 插值的概念，认为节点之间的面中心的点可以象节点那样的方式进行离散，就可以避免压力的振荡。

2)完全耦合的方法(Fully Coupled Method)

对人工压缩性的方法（Artificial Compressible Method）常常采用完全耦合的方法的进行求解。这种方法把不压缩流体看成是虚拟可压缩的，在流体速度散度为零的方程上加上一项压力对虚拟时间的导数项（乘以一人工压缩参数），形成新的压力方程。把待解的各个不同变量（压力，速度等）的未知数写成矢量的形式，接着对压力方程和动量方程联立离散后，再建立系数矩阵求解。把方程写成余差的形式，采用特征值基础上的通量分裂和隐式线性化的方法就是一种离散方法。完全耦合的方法往往收敛速度很快，但人工压缩参数需要合理选取，系数矩阵会占据较大的内存，并行计算也是一个考验。

2.6 大型线性方程组（稀疏矩阵）的求法

大型线性方程组的求法通常分为直接法和迭代法。而迭代法又可分为定常迭代法（Stationary iterative methods）法和Krylov子空间迭代方法。直接法和迭代法比较优点是稳定，缺点是需要更大的存储空间和更多的计算量，费内存费时间，并行计算也不易。求解大规模线性方程组一般不会采用直接法。定常迭代法容易实现，但通常效果不好（收敛速度慢），Krylov法是相对新的方法，虽然相对不易理解，但效果普遍优异（速度快）。

1)直接法（Direct Method）

比如高斯消去法（ Gaussian Elimination）。

2)定常迭代法（Stationary Iterative Methods, 举几个例子）

这种方法的迭代解不断在迭代过程中更新收敛，其他量则保持不变。

(1) Jacobi 法

Jacobi 法每迭代一次只需计算一次矩阵和向量的乘法，算法简单，容易实现并行计算。然而收敛速度较慢，占据的存储空间较大，所以工程中一般不直接用该法。

(2) Gauss-Seidel 法

Gauss-Seidel 方法与 Jacobi 方法类似，不同的是它使用更新后的值。一般来说，如果 Jacobi 方法收敛，Gauss-Seidel 方法将比 Jacobi 方法收敛得更快，尽管仍然相对较慢。

(3) SOR(Successive Overrelaxation)法

连续超松弛（SOR）可以通过引入外推参数从Gauss-Seidel方法导出。最佳选择的SOR的收敛速度比Gauss-Seidel要快一个数量级。

(4) ADI (The Alternating Direction Implicit method)

交替方向隐式法（ADI）方法是求解大型稀疏矩阵线性方程组的一种流行方法，适用于求解抛物型和椭圆型偏微分方程。这种方法往往用于结构网格的方法，可以实现具有三对角和五对角矩阵（在三个坐标方向上轮流迭代）的线性系统的快速精确的求解。

3) Krylov 子空间迭代法

Krylov子空间法目前被认为可用于求解大型线性系统的最重要的迭代技术。这些技术是基于Krylov子空间上的投影过程，包括正交投影和斜投影。

(1) 共轭梯度法（CG：The Conjugate Gradient Algorithm）

共轭梯度法是迭代的残差向量和 Krylov 子空间的向量相正交的方法。当系数矩阵是对称正定矩阵时，CG 是一种非常有效的方法，因为只需要存储有限数量的向量。

(2) 双共轭梯度法(BiCG : BiConjugate Gradient)

双共轭梯度法生成两个类似 CG 的向量序列，一个基于具有原始系数矩阵的系统，另一个基于他的转置矩阵。这种方法不是将每个序列正交化，而是使它们相互正交，或“双正交”。这种方法和 CG 一样使用有限的存储空间。它能处理矩阵是非对称和非奇异的情况，但收敛可能是不规则的，并且有可能该方法会崩溃。

(3) 稳定双共轭梯度法(Bi-CGSTAB :BiConjugate Gradient Stabilized)

稳定双共轭梯度法是 BiCG 的一个变种，是为了获得更快和更平滑的收敛性。

(4) GMRES(the Generalized Minimum Residual Method)：

GMRES 法利用 Krylov 子空间中的向量获得逼近解使得方程残差最小。GMRES 适合求解非对称系统。

4) 预处理（Preconditioning）技术

预处理技术是 Krylov 子空间方法在应用中取得成功的关键因素，好的矩阵的预处理方法可以有利于加快收敛速度。预处理的构造方法是一个很大的研究领域。

5) 多重网格（Multi-grid）方法

多重网格方法分成几何和代数的两种方法。几何的方法是把网格划分成粗细不同等级的几套分层网格系统，在粗细网格上依次计算和插值，利用网格的不同尺度快速降低不同频率的计算误差，从而加快计算的收敛速度。代数的多重网格法则仅利用系数矩阵的代数信息构造多重网格算法。代数的多重网格法对复杂几何形状和非结构网格问题的求解更有优势。

2.7 高性能计算（HPC: High Performance Computation）

高性能计算是利用超级计算机来计算标准的计算机来说太大量或者计算耗时太长的问题。一台台式机一般是一个处理芯片，即一个 CPU。高性能计算系统本质上是一个很多节点组成的网络，每个节点都包含一个或多个 CPU（每个 CPU 可以包括多个 core 或者 thread）以及自己的内存。

高性能计算采用各种并行计算的技术来提高计算效率（提高计算速度和合理分配内存）。并行计算是同时使用多个计算资源来解决一个计算问题，是把程序分成许多更小的处理器

（core）或线程（thread），每个部分的指令在不同的处理器（对 MPI 来说是进程）或线程上同时并行执行，因此提高计算速度。为了将较大的程序拼接在一起，处理器之间必须能够有效地相互通信，并且系统作为一个整体必须组织得很好。可扩展性（scalability）是检验并行效率一个很重要的指标。

1) Vectorization

矢量化（Vectorization）是实现并行性的一种特殊形式，比如 SIMD vectorization （core level）就是使用专用 SIMD（Single Instruction Multiple Data ）执行硬件单元，这些硬件单元位于使用特殊指令的处理器中。准备在向量处理器上使用程序的过程称为向量化（Vectorization）或 SIMDization。它可以由程序员手动完成，也可以由执行自动矢量化的编译器完成。SIMD 的处理过程是利用数据级（data-level）并行性。数据级并行意味着转换一组向量元素所需的操作可以同时在向量的所有元素上执行。也就是说，一条指令可以并行地应用于多个数据元素。

2) MPI (Message Passing Interface)

MPI（distributed memory , cluster level(inter- and intra- node parallelism)）是由学术界和工业界的一群研究人员（从事并行计算体系构造）设计的一种标准化的、可移植的消息传递标准。该标准定义了一个核心的库例程的语法和语义，它适用于广泛的用户编写 C、C++和 FORTRAN 中的可移植消息传递程序。MPI 有几种经过良好测试的高效实现，其中许多是开源的或用于公共领域。这些措施促进了并行软件产业的发展，并鼓励开发可移植和可扩展的大规模并行应用程序（摘自 Wikipedia）。

MPI是一个库，是一种标准或规范的代表，一种消息传递编程模型，往往用于分布存储并行机（distributed memory，但一个node内部可以是共享式内存），可以实现进程（processor）各自独立的并行计算和进程之间的消息传递，从而提高计算的效率。其目标是实现高性能，可扩展性和可移植性。 MPI 主要历经了MPI-1，MPI-2，和MPI-3三个版本，逐渐提高或添加新的功能。MPI的应用中要尽量用非阻塞通信方式，以提高并行效率。

3) OpenMP

OpenMP（shared memory, node level(thread parallelism)）是用于并行编程的库，使用OpenMP编程时，所有线程共享内存和数据。OpenMP支持C、C++和FORTRAN。OpenMP程序有顺序段(sequential)和并行段(parallel)。

4) CUDA(GPU, accelerator)

CUDA 是 NVIDIA 为图形处理单元（GPU）上的通用计算开发的并行计算平台和编程模型。有了 CUDA，开发人员可以通过利用 GPU 的强大功能大大加快计算应用程序的速度。在 GPU 加速的应用程序中，工作负载的顺序（sequential）部分在 CPU 上运行（针对单线程性能进行了优化），而应用程序的计算密集部分在数千个 GPU 内核上并行（parallel）运行。在使用 CUDA 时，开发人员使用流行语言如 C、C++、FORTRAN、Python 和 Matlab 进行编程，并通过扩展几个基本关键字的形式来表示并行性。NVIDIA 的 CUDA 工具包提供了开发 GPU 加速应用程序所需的一切。CUDA 工具包包括 GPU 加速库、编译器、开发工具和 CUDA 运行时（摘自 nvidia 官方网站）。

5)OpenACC （accelerator）

OpenACC 是一种基于用户驱动指令的性能可移植并行编程模型。它是为那些对将代码移植到各种异构 HPC 硬件平台和体系结构感兴趣的科学家和工程师设计的，其编程工作量要比低级别模型所需的少得多。OpenACC 规范支持 C、C++、FORTRAN 编程语言和多个硬件架构，包括 x86 和功率 CPUs，以及 Nvidia GPU（摘自 OpenACC 官方网站）。

6)混合并行计算方法

为了取得更好的计算效率，可采用各种混合并行计算方法，比如 MPI+X，X=OpenMP and/or OpenACC + Fortran or C/C++.

7) 区域分解方法（Domain Decomposition (DD) Method）

随着并行计算技术的稳步发展，DD 方法无疑是很著名的，也许是最有前途的类型之一。这中方法结合了偏微分方程的思想，线性代数、数学分析和图论技术。在偏微分方程的数值求解邻域，DD 方法是指将边值问题分解为子域上的较小的边值问题，并通过迭代协调邻域之间的解的值来解决边值问题。DD 方法比较典型的是作为 Krylov 子空间迭代方法（比如 GMRES, BiCG）的预处理器。

在偏微分方程的数值解法上，DD 方法可分为重叠（Overlapping decomposition）和非重叠（Non-overlapping decomposition）两种情况。重叠的技术比如 Schwarz formulation，非重叠的技术有比如 Steklov-Poincar'e formulation，Lagrange multiplier formulation 等，Least squares-control formulation 则既可用于重叠也可用于非重叠的情况。