编号 ××××××××

南京航空航天大學

本科毕业设计（论文）

题　目　基于梯度强化人工神经网络的翼型气动力优化

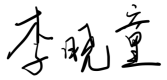
学生姓名	李晓童
学　　号	181910928
学　　院	航空学院
专　　业	飞行器设计与工程
班　　级	0119111
指导教师	高宜胜

二〇二三年六月

南京航空航天大学

本科毕业设计（论文）诚信承诺书

本人郑重声明：所呈交的毕业设计（论文）是本人在导师的指导下独立进行研究所取得的成果。尽我所知，除了文中特别加以标注和致谢的内容外，本设计（论文）不包含任何其他个人或集体已经发表或撰写的成果作品。对本设计（论文）所涉及的研究工作作出贡献的其他个人和集体，均已在文中以明确方式标明。

作者签名： 

日　　期：2023 年 06 月 12 日

南京航空航天大学

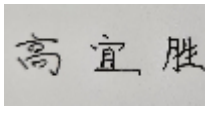
毕业设计（论文）使用授权书

本人完全了解南京航空航天大学有关收集、保留和使用本人所送交的毕业设计（论文）的规定，即：本科生在校攻读学位期间毕业设计（论文）工作的知识产权单位属南京航空航天大学。学校有权保留并向国家有关部门或机构送交毕业设计（论文）的复印件和电子版，允许论文被查阅和借阅，可以公布论文的全部或部分内容，可以采用影印、缩印或扫描等复制手段保存、汇编论文。保密的论文在解密后适用本声明。

论文涉密情况：

√ 不保密

□ 保密，保密期（起讫日期：　　）

作者签名：  　　导师签名： 

日　　期：2023 年 06 月 12 日　　日　　期：2023 年 06 月 12 日

摘　　要

围绕翼型优化设计问题的实际需要，使用梯度强化神经网络作为代理模型取代 CFD 仿真计算，对翼型的气动力参数进行预测，并进行了气动力优化。首先通过模型问题对比测试了传统神经网络，梯度强化神经网络和改进的梯度强化神经网络，显示了改进的梯度强化神经网络在预测性能和收敛速度方面的优越性。对于翼型数据预处理，使用奇异值分解对翼型进行参数化表达，将翼型分解为模态与模态系数。利用反距离加权插值方法构造约束函数，对模态系数进行限制，排除非正常翼型。采用拉丁超立方采样生成翼型样本，并利用 ADflow 计算气动力参数及其导数。计算结果用于训练改进的梯度强化神经网络从而得到预测翼型气动力的代理模型，并耦合优化软件包进行优化设计。优化结果与基于高置信度 CFD 的优化方法结果几乎一致，而耗时大大降低。

关键词：神经网络，梯度，翼型优化，奇异值分解

ABSTRACT

Focusing on the actual needs of airfoil optimization design problems, this paper proposes to use improved gradient-enhanced neural network as a surrogate model instead of CFD simulation calculation to predict the aerodynamic parameters of airfoils. By comparing and testing the traditional neural network, the gradient-enhanced neural network and the improved gradient-enhanced neural network in this paper, the superiority of the network in terms of prediction performance and convergence speed is clarified. For the preprocessing of data, this paper uses singular value decomposition to parameterize airfoils and decompose airfoils into modes and modal coefficients. The inverse distance-weighted interpolation method is used to construct a constraint function to limit the modal coefficient and exclude abnormal airfoils. Latin hypercube sampling was used to generate airfoil samples, and ADflow was used to calculate the aerodynamic parameters and their derivatives. The results are used for the training of the improved gradient-enhanced neural network to obtain a surrogate model for the prediction of the aerodynamics, and the neural network is coupled with the optimization software package for airfoil aerodynamic optimization. Compared with the high-fidelity CFD based optimization method, the optimization results are almost indistinguishable, but the optimization time is greatly reduced.

KEY WORDS: Artificial neural network, Gradient, Airfoil optimization, SVD

目录

第一章 绪论

1.1 背景和意义

随着民用航空的市场需求不断提高，人们对飞行器的飞行效率、经济性与安全性提出了越来越高的要求，因此越来越多的科研资金正在被投入到飞行器设计学科的研究当中。其中，翼型设计在飞机设计中扮演着重要的角色。机翼、固定翼飞机的水平和垂直尾翼以及直升机中的螺旋桨叶片，都是使用翼型横截面设计的。在早期，受限于计算能力，翼型气动数据一般只能通过风洞试验获得，翼型设计也一般采用风洞实验进行试凑比对，最终选择出一个最优外形。这些实验结果可靠，但比较昂贵且耗时。例如波音公司研制 B767，花费了 35000 小时进行风洞实验，空客公司研制 A310 进行了 18000 小时的风洞试验[1]，而 Northrop 公司研制 YF-17 进行了 13500 小时的风洞试验[2]。

而在最近的几十年里，随着计算流体力学的快速发展和计算机硬件水平的迅猛提升，计算空气动力学越来越多地应用于翼型设计，并借助数值优化来自动找到最优设计。翼型的设计问题转化为了一个优化问题。这种方法可以大大减少原本风洞实验所需的成本，以及缩短飞行器的设计周期，并且获得性能更好的气动外形。因此基于 CFD 的优化方法在飞行器设计中正在扮演越来越重要的角色[3]。

气动优化设计流程一般分为四个步骤[4]:第一，建立设计对象的优化设计模型（设计变量，目标函数和约束条件）；第二，对几何外形进行参数化；第三，气动性能计算；第四，使用优化搜索算法改进设计[3]。其中，第四点的优化搜索算法是气动优化设计的关键，关系到优化设计的效率和设计质量。

与传统 CFD 算法相比，基于代理模型的优化算法因为其计算代价远小于完全采用高置信度方法的优化策略，计算效率高，如今被越来越广泛地使用。代理模型的核心思想是利用简单易算的模型对设计空间进行参数化，并利用这个模型为优化算法生成输入[5]。这种模型通常被称为系统的响应面，从而定义了基于代理模型的优化方法[6]。在飞行器设计领域，基于代理模型的优化设计方法在结构优化设计[7]中最早使用，而后因为多学科设计优化（MDO）[8,15]的兴起而开始流行，并开始运用到气动优化设计[9,10]中去，发展出了多项式响应面（PRSM）[11,15]、Krigging 模型[12,15]、径向基函数（RBF）[13,15]、人工神经网络（ANN）、支持向量回归（SVR）[14]等多种形式的代理模型[3]。

很多常用的优化算法（如遗传算法和蚁群算法）不使用梯度信息，其计算量相当大，通常与设计变量维度成二次甚至三次相关。而随着伴随方法的发展，梯度的计算变得空前方便，利用梯度信息进行优化计算成为了可能[16,17]。本研究利用改进的梯度强化人工神经网络（mSANN）作为数值计算的代理模型，目标是减轻翼型优化设计的计算代价，缩短设计周期。很多现有研究是针对某个特定翼型进行优化设计，而本研究的目标是利用改进的梯度强化人工神经网络建立一个适用于亚音速和跨音速条件下较多种类翼型的优化设计工具，能够较为精确快速地获得翼型在特定流动条件下的最优外形。

1.2 国内外研究现状

如今，由于计算资源和训练所需的数据集获取门槛降低，人工神经网络也逐渐普及开来[18]。人工神经网络最早是由生理学家 Mc Culloch 和数学家 Pitts 在 1943 年提出，他们把人工神经网络 ANN 描述为一个类似人类大脑的计算模型（M-P 模型），将阈值函数作为计算神经元的主要特性，标志了人工神经网络的诞生[19, 20]。1949 年，心理学家 Donala O.Hebb 在 The organization of behavior 中提出了 Hebb 突触和连接权值强化的 Hebb 法则：在神经网络中，信息存储在连接权中，神经元之间突触的联系强度是可变的，这种变化建立起神经元之间的连接[21]，为人工神经网络算法的发展构建了理论知识基础[22]。上世纪 60 年代末，Rosenblatt 提出“感知器”模型。感知器是第一个物理构建并形成具有学习能力的人工神经网络，其建立在 M-P 模型的基础上[19]。1984 年，Hopfield 神经网络（Hopfield Neural Network, HNN）被首次提出[23]。为了解决更加复杂的问题，需要增加神经网络的层数，反向传播网络（Backpropagation Neural Network, BPNN）被提出用于解决多层神经网络的问题[24]。但是 BP 网络仍存在一些缺点，比如收敛速度慢，以及大样本数据收敛困难，易出现局部最小化的现象[19]。1998 年，基于福岛邦彦提出的卷积和池化结构，LeCun 等人将 BP 反向传播算法运用到该结构的训练中，便获得了卷积神经网络（Convolutional Neural Network, CNN）的雏形 Le Net-5[25]。目前，人工神经网络 ANN 被应用于许多热门的研究主题，如自动驾驶汽车（Angelova 等人，2015）[26]、计算机视觉（Ba 等人，2014 年）[27]和语音识别（Heigold 等人，2013 年）[28]。

人工神经网络也被广泛应用于航空航天领域。2000 年，Rai 和 Madavan[18, 29]使用 ANN 和多项式模型来近似具有 15 个设计变量的亚音速流动中涡轮叶片的压力分布，一开始在优化过程中动态调整了神经网络，非常耗时。后来为了在准确性和效率之间权衡，他们选择在优化的初始阶段使用基于欧拉方程数据的代理模型，而在最后阶段切换到 RANS 数据。

2018 年，Chen 和 Agarwa[30,31]在研究风力发电机的扇叶优化设计问题时指出利用 ANN 和遗传算法相结合时，计算效率大大提高，并能获得全局最优解。2013 年，西安理工大学的朱国俊等[32]采用 RBF 径向基神经网络与 NSGA-II 遗传算法相结合的方法解决水力翼型的多工况优化问题。对 NACA63-815 翼型进行了优化改进, 重点研究该翼型在 3 个攻角工况下 (0, 6° 和 12°) 的优化问题。2017 年，浙江大学的张玄武[33]采用类别形状函数 (CST) 参数化方法，对翼型进行参数化，并随机生成翼型样本对级联前向神经网络进行训练，并将其作为翼型流场数值计算的代理模型。试验结果表明，用级联前向网络计算出的升阻比能够达到需要的精度要求，同时对于给定的优化目标可节约大量计算时间。2021 年，上海理工大学的陈晨铭等[34]采用 BP 神经网络和遗传算法相结合的方法对 NACA64(3)-618 风力机翼型进行了气动优化。他们利用拉丁超立方采样生成样本，并通过 B 样条曲线对翼型进行光滑化处理，再结合遗传算法实现气动优化选型，最后利用 CFD 方法验证了优化结果。

ANN 已扩展到多信息源建模。然而，使用梯度信息来解决工程问题的 ANN 研究还较为有限。Giannakoglou 等[35]使用梯度强化的 ANN 分析了一些 2D 和 3D 空气动力学形状设计问题。他们以函数误差和梯度误差的组合作为误差指标，使用了欧拉方程模型，并只考虑了马赫数为 0.5 的情况。

Liu 和 Batill[36]开发了一种梯度强化的 ANN 方法，在他们的方法里没有对目标和梯度的残差进行加权，但他们只解决了最多 15 个输入变量的分析基准。Pan 和 Duraisamy[37]使用 ANN 对具有梯度信息的非线性动力系统进行建模。在该研究中，梯度作为罚函数项，以此获得平滑响应，而不是将 ANN 模型的梯度与数据的梯度相匹配。2017 年，Czarnecki 等[38]开发了另一种使用梯度强化 ANN 的方法，其中损失函数被定义为输出估计误差和导数估计误差的加权和。2020 年，密歇根大学的 Martins 等[18]在翼型优化中应用了这种将输出误差和导数误差的加权和作为损失函数的梯度强化神经网络的方法，将其作为气动力计算的代理模型，并与西北工业大学李记超博士[39]在 2019 年使用的偏最小二乘法梯度强化 Kriging 模型（GE-KPLS）进行了对比，比较两者对翼型升力系数，阻力系数，以及力矩系数的预测表现。本文的研究内容主要基于对以上成果的学习，力求在真实复现的基础上，根据自己的理解简化其使用。

1.3 本文主要工作

针对传统优化算法进行翼型优化耗时较长的问题，本次研究旨在开发一个适用于亚音速和跨音速条件下较多种类翼型的优化设计工具，能够较为精确快速地获得这些翼型在特定流动条件下的最优外形。

本次研究从选择适合翼型形式构建翼型库开始，通过对弯度和厚度进行 SVD 分解（奇异值分解）实现翼型参数化；利用拉丁超立方采样（LHS）方法生成翼型样本和流动状态，以作为训练样本；采用开源 CFD 软件 ADflow 计算不同迎角、Mach 数条件下翼型的气动力参数，并利用 ADflow 的离散伴随功能计算气动力参数对于迎角、Mach 数的梯度；使用气动力参数和相应的梯度信息作为训练样本，利用机器学习框架 TensorFlow 构造改进的梯度强化神经网络（mSANN）并进行训练；以改进的梯度强化神经网络作为代理模型耦合梯度优化软件包 SNOPT 进行翼型优化设计。

1.4 论文组织结构

本论文的组织结构如下：

第一章 阐明本次研究的研究背景，介绍课题的研究现状，并概述论文的主要工作和创新点；

第二章 解释梯度强化神经网络的定义与结构，测试效果和训练方法；

第三章 介绍如何使用翼型参数化方法定义设计空间，如何定义设计变量和流动状态的边界，并利用拉丁超立方采样进行采样，生成样本集，利用 ADflow 进行气动力计算；

第四章 展示利用样本集训练梯度强化神经网络的效果，并耦合 SNOPT 优化软件包进行优化设计，以及与 CFD 优化方法的效果对比；

第五章 总结全文，对今后的研究做出总结和展望。

第二章 梯度强化神经网络

2.1 梯度强化神经网络的定义

本研究所采用的改进梯度强化神经网络是密歇根大学的Martins等[18]基于Czarnecki等人[38]开发的梯度强化神经网络方法的改进版本。Czarnecki等人提出了一种ANN的Sobolev训练方法（Sobolev training for artificial neural networks, SANN)，而Martins团队为了提高SANN的训练效率，在其损失函数中引入了一个权重系数，大大提高了网络的收敛速度。下面将对这两种网络进行一个简单的描述。

2.1.1 SANN的定义

为了训练一个神经网络，需要建立训练集输入矩阵：

$$\boldsymbol{X} = \left[X^{(1)}, \cdots, X^{(n_t)}\right]^T, \tag{2.1}$$

每一个训练样本为：

$$X^{(i)} = [x_1^{(i)}, \cdots, x_d^{(i)}]^T, \tag{2.2}$$

那么输入矩阵 $\boldsymbol{X}$ 是一个（$n_t \times d$）的矩阵，d是输入样本的维度，n_t是输入样本的数量。与之相对应，网络的训练输出格式是：

$$\boldsymbol{y} = [y^{(1)}, \cdots, y^{(n_t)}]^T, \tag{2.3}$$

这里定义SANN想要拟合函数f的功能，则有$y^{(i)} = f\left(X^{(i)}\right)$，那么网络的每个输出结果对输入的梯度可以表示为：

$$\nabla f\left(X^{(i)}\right) = [\frac{\partial f}{\partial x_1^{(i)}}\left(X^{(i)}\right),\ \cdots,\ \frac{\partial f}{\partial x_d^{(i)}}\left(X^{(i)}\right)]^T, \tag{2.4}$$

整个训练的梯度矩阵可以表示为：

$$\mathbf{df} = \nabla f(X) = [\nabla f\left(X^{(i)}\right), \cdots, \nabla f\left(X^{(nt)}\right)]^T\ , \tag{2.5}$$

在神经网络的训练过程中，采用一系列训练参数θ，通过修改参数θ来使网络的损失函数最小，损失函数可表示为：

$$\min_{\theta} y_{loss}(\boldsymbol{X}, \boldsymbol{y}|\theta),$$

$$y_{loss}(\boldsymbol{X}, \boldsymbol{y}|\theta) \coloneqq l\left(m\left(x^{(i)}\middle|\theta\right), y^{(i)}\right), \tag{2.6}$$

其中l表示损失函数。

为了将梯度信息引入神经网络的训练中，Czarnecki 等人[38]在 SANN 的损失函数中引入了包含梯度的项，如下所示：

$$\min_{\theta} y_{loss}(\boldsymbol{X},\boldsymbol{y},\mathbf{df}|\theta),$$

$$y_{loss}(\boldsymbol{X},\boldsymbol{y},\mathbf{df}|\theta) \coloneqq l\left(m\left(x^{(i)}\middle|\theta\right),y^{(i)}\right) + \sum_{j=1}^{d} l_j\left(\frac{\partial m}{\partial x_j^{(i)}}\left(X^{(i)}|\theta\right),\frac{\partial f}{\partial x_j^{(i)}}\left(X^{(i)}\right)\right), \quad (2.7)$$

其中l_j表示第 j 个偏导数的损失函数。通过公式 2.7 可以看出，第二项关于梯度的损失函数项占据总损失函数的主要部分，尤其是当训练样本的维度 d 很大的时候，由于要追求梯度误差的最小化，网络的损失函数下降会很缓慢，这会导致 SANN 的训练十分耗时。

2.1.2 mSANN（modified SANN）的定义

为了加快神经网络的收敛速度，Martins 团队 [16] 对 SANN 的损失函数进行了改进，对第二项梯度损失项引入了一个权重系数λ_k：

$$\min_{\theta} y_{loss}(\boldsymbol{X},\boldsymbol{y},\mathbf{df}|\theta),$$

$$y_{loss}(\boldsymbol{X},\boldsymbol{y},\mathbf{df}|\theta) \coloneqq \sum_{i=1}^{n_t} l\left(m\left(x^{(i)}\middle|\theta\right),y^{(i)}\right) + \lambda_k \sum_{j=1}^{d} l_j\left(\frac{\partial m}{\partial x_j^{(i)}}\left(X^{(i)}|\theta\right),\frac{\partial f}{\partial x_j^{(i)}}\left(X^{(i)}\right)\right), \quad (2.8)$$

其中$\lambda_k \in [0,1]$，作用是调整梯度项对总损失函数的影响。该方法的好处是可以在训练中逐渐引入梯度信息的作用，既能提高网络的预测精度，也不会耗费大量时间。

2.1.3 神经网络的结构

本研究所使用的网络结构参考了 Martins 教授团队的研究结果。使用 NN- SVG 绘图工具绘制了神经网络的结构图。网络一共有 8 层，包含 1 个输入层和 1 个输出层，以及 6 个全连接层，每个全连接层有 100 个节点。为了便于测试对比三种网络的性能，把输入层统一成 16 个节点，输出层 1 个节点。前两层全连接层的激活函数为双曲正切函数，第三和第四层的激活函数为 Sigmoid 函数，第五层的激活函数为 Leaky Relu 函数，第六层的激活函数为 Relu 函数。结构如图 2.1 所示。

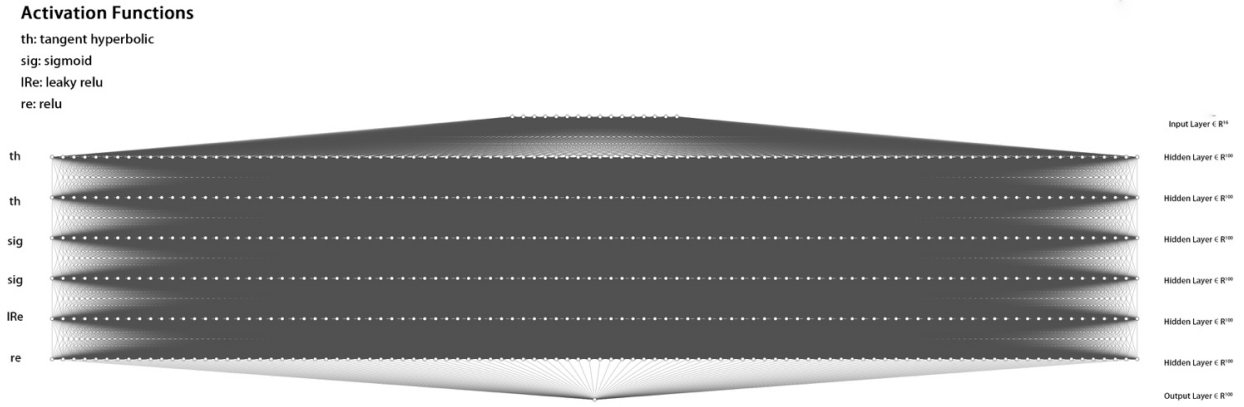


图 2.1 神经网络结构图

2.2 梯度强化神经网络的性能

2.2.1 测试函数的选择

为了验证 mSANN 的性能优势，对按上一节所描述的结构，利用开源机器学习框架 Tensorflow 搭建了 ANN，SANN，以及 mSANN，选取 Rosenbrock 函数对比它们的性能。该函数形式如下：

$$\sum_{i=1}^{15}[(x_{i+1}-x_i^2)^2+(x_i-1)^2], \qquad -2 \le x_i \le 2,$$

$$\text{for i=1,}\cdots\text{,16.} \tag{2.9}$$

对于 mSANN，其超参数$\lambda_k = 0, 0.1, 0.2, \cdots, 1.0$。

2.2.2 性能测试结果与对比

测试数据集来自密歇根大学 MDO Lab 的开源数据集，包含 42000 个训练样本和 22000 个验证样本，测试结果如下图所示：

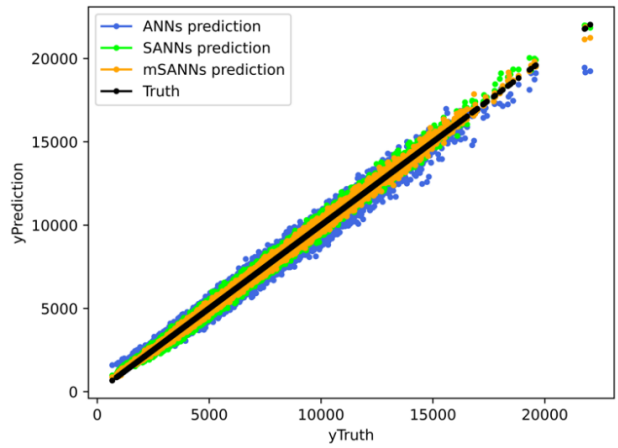


图 2.2 ANN，SANN，mSANN 的预测能力

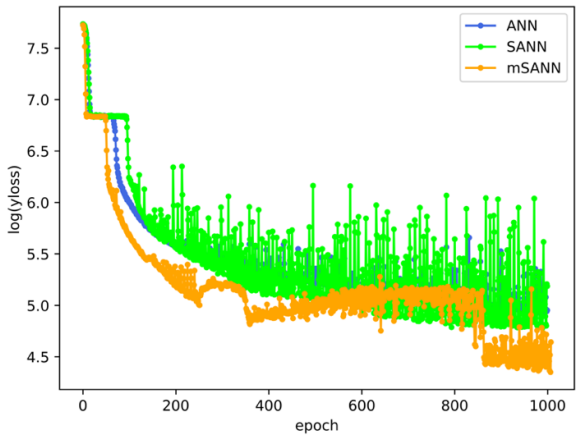


图 2.3 ANN，SANN，mSANN 的收敛速度

图 2.2 展示了三种神经网络的预测能力，可以看出 mSANN 的预测精度最高，其次是 SANN。

图 2.3 描绘了三种网络的损失函数收敛速度，可以看到，得益于引入了权重系数λ_k，mSANN 与 SANN 相比，收敛速度大大提升，残差也更小。

接着对三种神经网络的预测结果计算了 L_2 范数，对于函数值y的预测结果，L_2 范数为：

$$\epsilon_{L_2} = \frac{||\hat{y}-y||_2}{||y||_2}100, \tag{2.10}$$

其中，y是一个向量，包含了函数对应输入（训练集和测试集）的真实值，$\hat{y}$也是一个向量，包含了神经网络对函数的预测值。L_2 范数的数值越小，则说明预测结果越精准。

对于函数值的导数dy的预测结果，L_2 范数为：

$$\bar{\epsilon}_{L_2} = \frac{||\widehat{dy}-dy||_2}{||dy||_2}100, \tag{2.11}$$

表 2.1 展示了 L_2 范数的计算结果，表明不管是对输出的预测还是导数的预测，mSANN 的预测能力都要强于 ANN 和 SANN。

表 2.1 三种神经网络的 L_2 范数对比

	训练集		测试集	
	ϵ_{L_2}	$\bar{\epsilon}_{L_2}$	ϵ_{L_2}	$\bar{\epsilon}_{L_2}$
mSANN	2.05	8.06	2.12	8.11
SANN	2.27	8.48	2.30	8.49
ANN	4.35	24.79	4.70	24.92

第三章 数据预处理方法和 CFD 计算

3.1 翼型参数化方法

3.1.1 翼型数据选取

为了获得较大的翼型设计空间，训练用的的翼型数据应该尽量多且包含各种系列的翼型。本研究从伊利诺伊大学厄巴纳-香槟分校的 UIUC 翼型库中，选取了 1,389 个亚音速翼型和 21 个 NACA SC（2）系列的跨音速翼型，可以覆盖较大的设计空间。

3.1.2 翼型数据预处理

在建立包含 m 个翼型，每个翼型 n 个坐标点的翼型矩阵之前，首先要对翼型数据进行预处理。这是因为现有的翼型数据文件中，每个翼型所使用的控制点数量都不相同，数据形式多样，需要进行统一。本研究采用了 MDO Lab 开发的翼型处理工具 prefoil，可以对翼型数据进行插值拟合，将弦长归一化后，对翼型采用统一的采样规律进行采样，这样就可以获得 n 个 x 坐标相同的翼型坐标点。把这 m 个翼型数据整合到一个矩阵当中，就获得了如下所示的翼型矩阵：

$$A = \begin{bmatrix} y_{11} & y_{21} & \cdots & y_{m1} \\ y_{12} & y_{22} & \cdots & y_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{1n} & y_{2n} & \cdots & y_{mn} \end{bmatrix}, \tag{3.1}$$

3.1.3 奇异值分解

奇异值分解（Singular Value Decomposition，SVD）是线性代数里一种常见的分解算法，可用于数据降维，提取矩阵的主要特征，常常用于主成分分析算法（PCA），推荐算法或自然语言处理（NLP）当中。这里采用 SVD 算法，对翼型矩阵进行分解，提取翼型外形的模态特征，作为翼型参数化方法。

对矩阵 A 进行 SVD 操作：

$$A = U\Sigma V^T, \tag{3.2}$$

其中，矩阵U为$n \times n$的全翼型模态矩阵，矩阵Σ为$n \times m$的对角矩阵，包含了矩阵A的奇异值，矩阵V为$m \times m$的正交矩阵。其中，矩阵Σ中包含的A的奇异值是按由大到小依次排列，那么U中的模态也是按照对总特征的重要性排序的。由于奇异值的大小一般下降得非常快，通常不会用到所有的模态，而是只取前几个最重要的模态。如果用Φ来代表前n_f个想要考虑的模态，用a_{full}来表示模态系数，那么一个翼型的y坐标可以用下面的公式表示：

$$y = \Phi a_{full}, \tag{3.3}$$

图 3.1 是从 UIUC 翼型库 1389 个低速翼型提取的前 20 个翼型模态，这些模态代表着翼型的各种外形特征。大部分模态既包含弯度特征也包含厚度特征，而有的模态，比如 6 号，10 号，11 号模态，几乎只包含了弯度信息。

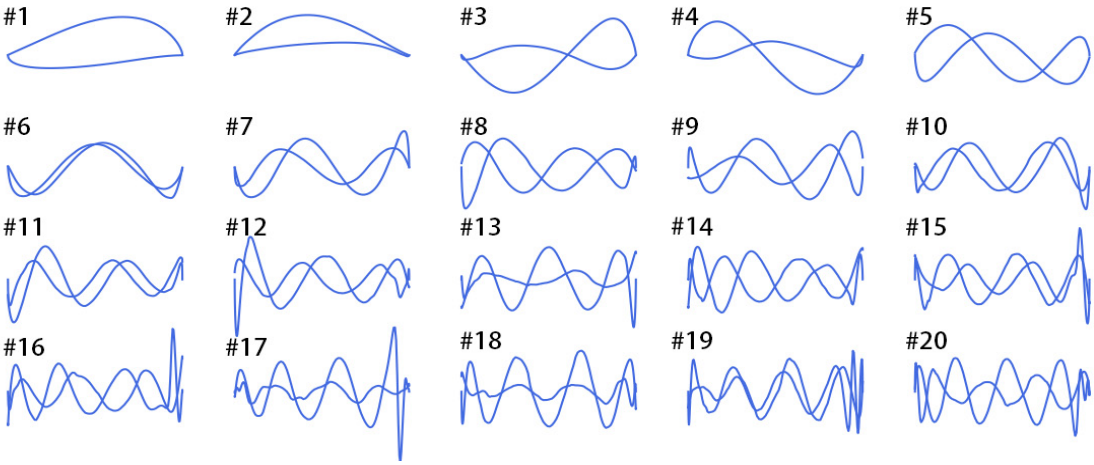


图 3.1 UIUC 翼型库 1389 个低速翼型的前 20 个翼型模态

但在本研究中，选择对翼型的弯度矩阵和厚度矩阵进行分解，而不是对翼型矩阵分解，因为这样可以更好地对翼型施加厚度约束。

弯度矩阵和厚度矩阵分别表示翼型的弯度线和厚度线的坐标，而且因为知道每个x坐标处的弯度和厚度后，该点上下表面的翼型坐标都能计算得出，故弯度矩阵和厚度矩阵中每个翼型的数据量是全翼型矩阵的一半：

$$A_c = \begin{bmatrix} y_{c_{11}} & y_{c_{21}} & \cdots & y_{c_{m1}} \\ y_{c_{12}} & y_{c_{22}} & \cdots & y_{c_{m1}} \\ \vdots & \vdots & \ddots & \vdots \\ y_{c_{1n_h}} & y_{c_{2n_h}} & \cdots & y_{c_{mn_h}} \end{bmatrix}$$

$$A_t = \begin{bmatrix} y_{t_{11}} & y_{t_{21}} & \cdots & y_{t_{m1}} \\ y_{t_{12}} & y_{t_{22}} & \cdots & y_{t_{m1}} \\ \vdots & \vdots & \ddots & \vdots \\ y_{t_{1n_h}} & y_{t_{2n_h}} & \cdots & y_{t_{mn_h}} \end{bmatrix}$$

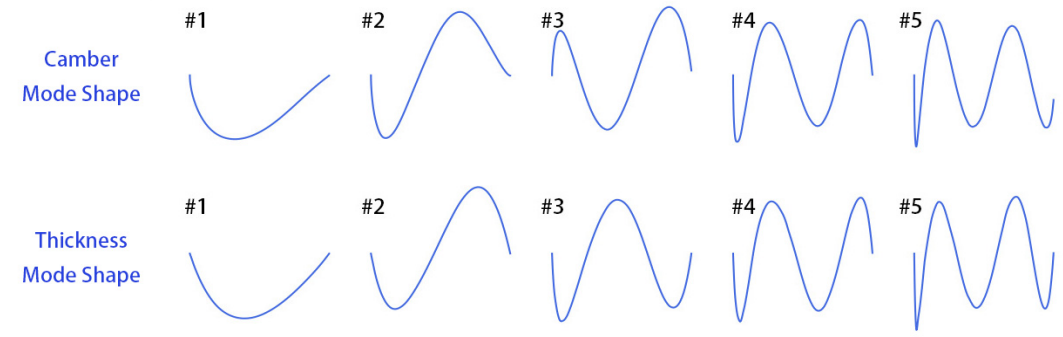
其中$n_h = n/2$。弯度模态和厚度模态同样可通过对A_c和A_t进行奇异值分解得到：

$$A_c = U_c \Sigma_c {V_c}^T, \quad A_t = U_t \Sigma_t {V_t}^T$$

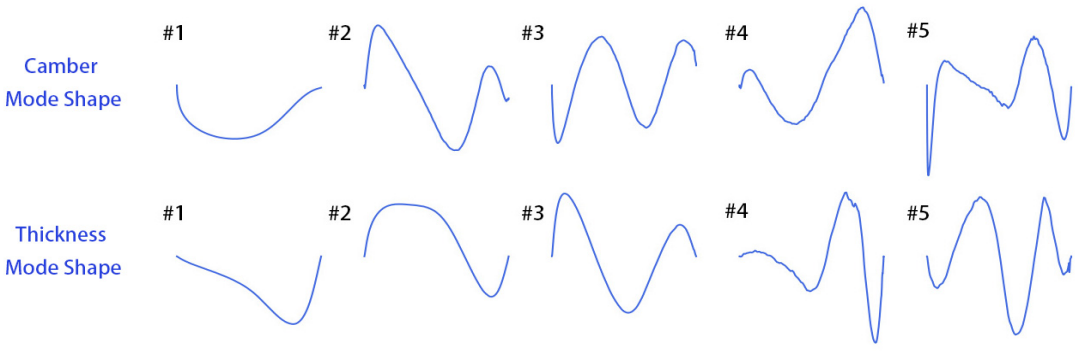
用Φ_c和Φ_t分别来代表前n_c个弯度模态和前n_t个厚度模态，用a_c和a_t来表示弯度和厚度模态系数。计算翼型 y 坐标的方法是先算出弯度线的坐标，再加上或减去半厚度，这样便可以获得翼型上下表面的的 y 坐标，用矩阵可以表示为：

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_{upper} \\ \boldsymbol{y}_{lower} \end{bmatrix} = \begin{bmatrix} \Phi_c & \Phi_t \\ \Phi_c & -\Phi_t \end{bmatrix} \begin{bmatrix} a_c \\ \frac{1}{2} a_t \end{bmatrix} = \Phi_{ct} a_{ct}, \tag{3.4}$$

从 UIUC 低速翼型库和 NACA SC（2）跨音速翼型库的翼型分解得到的前五个弯度模态和厚度模态如图 3.2 所示。



a)　1389 个 UIUC 低速翼型的前五个弯度和厚度模态



b)　21 个 NACA SC（2）跨音速翼型的前五个弯度和厚度模态

图 3.2　不同类型翼型的模态特征

3.2　样本的约束条件与生成

3.2.1　翼型模态数量选择和模态系数的约束边界

3.1 节对弯度和厚度矩阵进行了奇异值分解，分别获得了各自的一系列模态和对应的奇异值，奇异值的大小表明了该模态对翼型外形特征的重要性。因此接下来要考虑的问题就是如何选择合适数量的模态进行设计。这个问题需要找到一个最优平衡点，因为增加模态数量，一方面可以提高拟合翼型的准确度，但另一方面会加重神经网络的计算任务量，需要更长的训练时间，更容易出现过拟合现象；而减少模态数量也会有相应的优势与缺点。在先前的研究当中，人们发现对于二维翼型通常需要数十个设计变量，对于三维翼型则通常需要数百个设计变量，才能满足优化设计的目标[16]。李记超等[39]的实验结果表明，对亚音速翼型选取 7 个弯度模态和 7 个厚度模态作为设计参数，对跨音速翼型选取 4 个弯度模

态和 4 个厚度模态作为设计参数进行翼型设计，描述翼型的映射误差ϵ_p可以控制在 0.5%以内，对 1172 个利用 14 个模态设计的亚音速翼型进行了 RANS CFD 仿真计算，对阻力系数C_d的平均误差只有 0.53 计数，证明这样的选择（控制$\epsilon_p \leq 0.5\%$）是符合要求的。因此，本文对模态数量的选择参考了该研究，采用了同样的选择。

对于设计空间，一般的想法是设计空间越大越好。但是如果对模态系数不加限制的话，有很大的概率会得到不希望出现的奇怪翼型，因此要设计一个合适的约束边界，既能避免这些不期望的翼型，也能保留较大的设计空间。

初步考虑利用每个模态所对应的模态系数中的最大值和最小值来约束该列模态系数的范围，表示为：

$$\begin{cases} a_{c_{lower}}^{(j)} = \min a_c^{(j)} \\ a_{c_{upper}}^{(j)} = \max\ a_c^{(j)} \end{cases}, \quad \begin{cases} a_{t_{lower}}^{(j)} = \min a_t^{(j)} \\ a_{t_{upper}}^{(j)} = \max\ a_t^{(j)} \end{cases}, \tag{3.5}$$

其中$a_c^{(j)} \in \mathbb{R}^m$，$j = 1, \cdots, n_c$，$a_t^{(j)} \in \mathbb{R}^m$，$j = 1, \cdots, n_t$，代表了 m 个翼型各自包含的$n_c$个弯度模态和$n_t$个厚度模态。但是这种约束起到的效果十分有限，尤其是对于翼型数据较多的 UIUC 亚音速系列翼型，并不能很好地消除奇怪翼型。图 3.3 给出了一个在实验中用此约束方法采样得到的奇怪翼型。

图 3.3 对模态系数施加粗略约束得到的奇怪翼型

因此需要进一步限制模态系数的范围。由于一阶模态对总体外形特征做出的贡献最大，赋予最大的自由度，即对一阶模态系数的约束仍然按照公式（3.5）处理，这样可以保证设计空间的大小。对于高阶模态，想要在设计新翼型时考虑到高阶模态在原始翼型库中的分布与一阶模态的关系。在根据参考文件[39]，尝试构造关于a_{c_1}和a_{c_1}的函数$S(a_{c_1}, a_{t_1})$，函数值为高阶模态系数，具体方法为：

1) 利用现有翼型库的数据进行插值拟合得到$S_{c_i}(a_{c_1}, a_{t_1})$和$S_{t_j}(a_{c_1}, a_{t_1})$。这里采用反距离加权插值法，以$a_{c_1}$和$a_{t_1}$的插值作为区域的边界，这样便可以获得每个一阶模态系数取值点处，各个高阶模态系数的取值。

2) 构造边界约束函数$S_{upper} = \max(0, S)$，$S_{lower} = \min(0, S)$。

经测试，利用此方法对高阶模态系数进行约束后，基本避免了奇怪翼型，可以进行采样。

3.2.2 利用拉丁超立方采样方法生成样本

在训练神经网络之前，首先要生成大量的翼型样本，并对他们进行 CFD 计算气动力参数，以及利用离散伴随方法计算气动力参数对于模态的导数，作为训练的输入样本。对于亚音速翼型，其外形由 14 个模态（7 个弯度模态和 7 个厚度模态）控制，迎角范围$a \in$ [-2°, 6°]，马赫数$M \in [0.3, 0.6]$。假定流动状态的条件是基于 10000 米高度的标准大气，雷诺数 Re 变化的范围是$[2.4 \times 10^6, 5.0 \times 10^6]$。对于跨音速翼型，其外形由 8 个模态（4 个弯度模态和 4 个厚度模态）控制，迎角范围$a \in$ [-1.5°, 4.5°]，马赫数范围$M \in [0.7, 0.85]$，雷诺数范围 $Re \in [5.6 \times 10^6, 7.3 \times 10^6]$。

在上述的变量范围，以及 3.2.1 节中确定的一阶模态系数范围下，采用拉丁超立方采样（Latin Hypecube Sampling，LHS）来生成样本。拉丁超立方采样方法通过对变量范围预先等分成若干个子区域，再在每一个子区域中随机抽样，最后打乱组合，生成样本。这样可以保证样本不会发生集中现象，每一个子区域都会被抽样到，提高了样本的覆盖能力。最终采用 LHS 方法，生成了 2000 个亚音速翼型样本以及 2000 个跨音速翼型样本。

3.3 气动力参数计算和梯度计算

3.3.1 气动力参数计算

本次研究选择 ADflow 作为 CFD 求解器进行气动力参数的计算，采用 Spalart–Allmaras 湍流模型求解 RANS。利用 pyHyp 库编写 Python 脚本自动生成所有翼型的 CFD 网格。图 3.4 为使用 pyHyp 生成的不同翼型 CFD 网格。

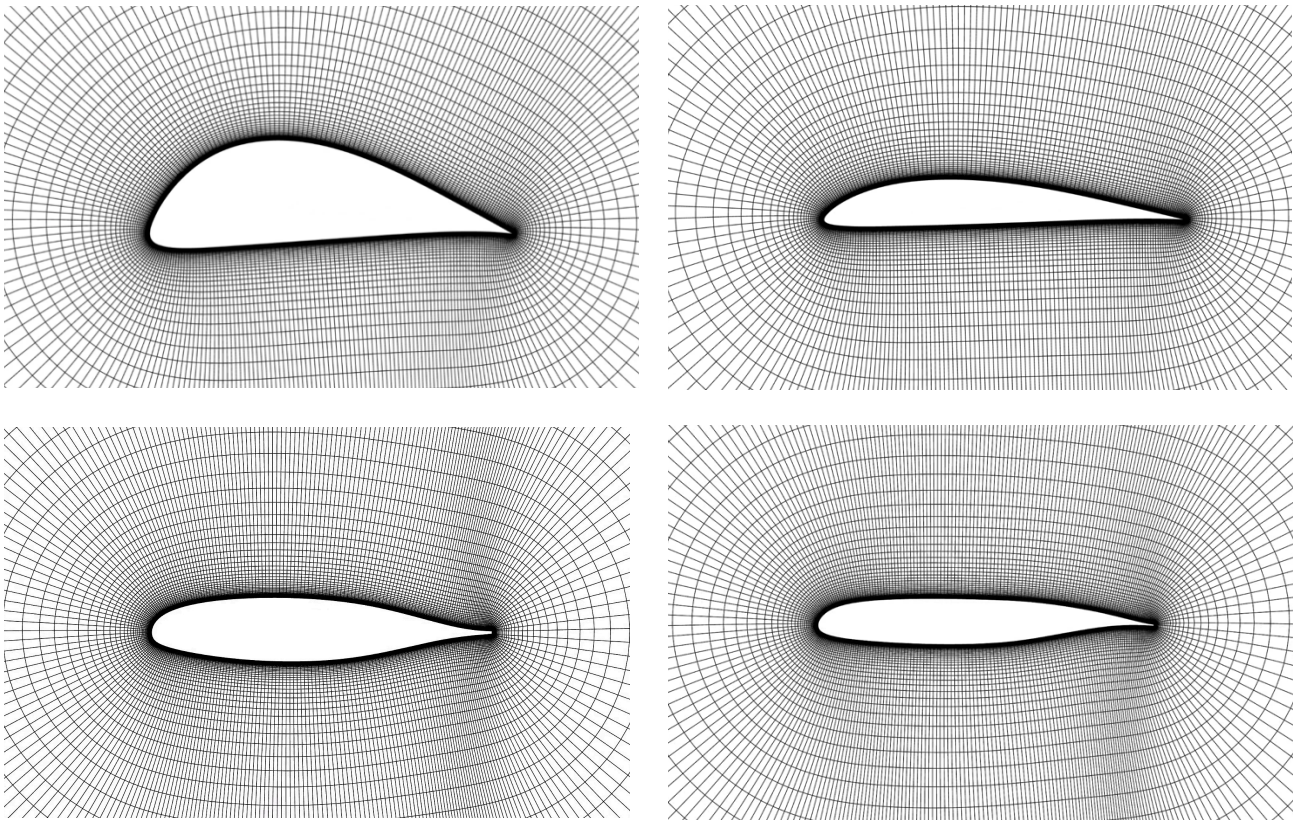


图 3.4 pyHyp 生成的不同翼型网格

3.3.2 梯度计算

准备训练数据的最后一步是获得目标函数关于设计变量的梯度，又称为灵敏度（sensitivity）。有限差分方法和解析方法是最早被应用于计算灵敏度的两种方法[40]。前者对于灵敏度的计算精度取决于扰动步长，后者通过求解线化方程得到灵敏度，精度较高。但是这两种方法的计算量都与设计变量的维数呈线性关系（而一些无梯度优化方法，如遗传算法，其计算量一般与设计变量维数的二次方甚至三次方相关[17]）。而伴随方法则是一种更高效的梯度求解方法，每一步求解过程只需要分别求解一次非线性流场和线性的伴随方程，计算量几乎与设计变量的数目无关，大大提高了大规模设计变量优化问题的计算效率。

本次研究采用的 ADflow 包含了离散伴随求解功能，可以高效地计算气动力参数对于翼型表面坐标，迎角和马赫数的梯度。

在对在 3.2 节中采样的翼型样本进行了计算之后，便得到了气动力参数对于翼型坐标的导数，但本研究中最终的设计变量是翼型模态系数而非翼型坐标。为了获得气动力参数对于模态系数的导数/梯度，需要计算出翼型坐标对于模态系数的导数/梯度，再根据链式法则得到气动力参数对于模态系数的导数/梯度，见下式:

$$\frac{\partial C_l}{\partial a_c} = \frac{\partial C_l}{\partial y} \times \frac{\partial y}{\partial a_c} = \left[\frac{\partial C_l}{\partial y_1} \quad \cdots \quad \frac{\partial C_l}{\partial y_n}\right] \begin{bmatrix} \frac{\partial y_1}{\partial a_c} \\ \vdots \\ \frac{\partial y_n}{\partial a_c} \end{bmatrix}, \tag{3.6}$$

至此，训练神经网络所需的数据集准备完成。

第四章 梯度强化神经网络的训练和翼型优化设计

4.1 梯度强化神经网络的训练

4.1.1 统一亚音速和跨音速训练样本的输入格式

由于亚音速翼型使用的是 7 个弯度模态和 7 个厚度模态，而跨音速翼型使用的是 4 个弯度模态和 4 个厚度模态，因此它们的训练样本的维度是不相同的。如果要为两个速域分别训练模型较为麻烦，更希望训练一个可以对亚音速和跨音速翼型都具有预测能力的统一模型，因此需要把这两种输入格式统一起来。

这里使用了 Martins 团队[18]的方法。构造一个稀疏矩阵，对于每个速域的输入，参数的矩阵格式为$n \times d$，其中n为样本的数量，d为每个样本数据的维度（包含了翼型模态系数和两个流动参数）。将输入记为一个矩阵X，把它分为模态系数部分和流动参数部分，则可表示为$X = [X_{ct} \quad X_{\alpha M}]$，之后将亚音速和跨音速数据融合放入该矩阵，行数$n = n_{sub} + n_{trans}$，列数$d = d_{sub} + d_{trans} - 2$，表示如下：

$$X = \begin{bmatrix} X_{ct_{trans}} & 0 & X_{\alpha M_{trans}} \\ 0 & X_{ct_{sub}} & X_{\alpha M_{sub}} \end{bmatrix}, \tag{4.1}$$

4.1.2 训练与验证

训练的流程如图 4.1 所示。首先将数据整合到 4.1.1 节中的矩阵当中，然后对 mSANN 进行迭代训练，若验证网络的预测能力没有达到要求，则继续训练；若达到要求，则跳出迭代。由于 2000 个样本的训练效果不佳，最终选用了李记超等[39]的开源数据，即 42000 个亚音速样本和 4000 个跨音速样本。

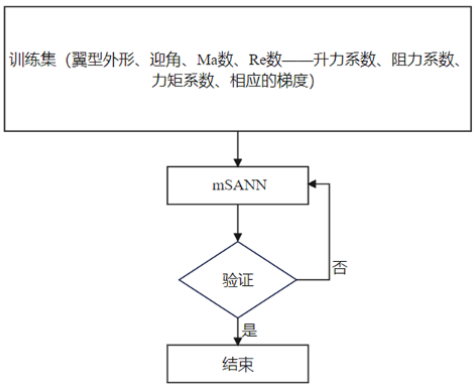


图 4.1 训练 mSANN 流程图

为了评估 mSANN 模型的性能，计算了验证和测试集的相对误差的 L_2 范数（式 2.10）。还计算了每个样本（包含验证和测试集）相对误差的 L_1 范数：

$$\epsilon_{L_1} = \left| \frac{y_{pred} - y_{true}}{y_{true}} \right| 100, \tag{4.2}$$

4.2 翼型优化设计

训练完成后，将获得的 mSANN 模型耦合梯度优化软件包 SNOPT（Sparse Nonlinear OPTimizer）对翼型进行优化。为了验证模型的效果，本文对亚音速和跨音速翼型分别选取了一种翼型进行优化测试。

4.2.1 亚音速翼型优化

对于亚音速翼型，本文选择 NACA0012 翼型作为测试范例。把马赫数固定在 0.45，以 14 个模态系数和迎角作为设计变量，并对升力系数设置约束为$C_l = 0.5$，对C_d进行最小化。为了避免非正常翼型，将一阶厚度模态控制在 NACA0012 的一阶厚度模态的 90%以下，其余 13 个模态系数则按照 3.2.1 节中的方法进行约束。

图 4.2 展示了 mSANN 方法和 CFD 方法进行优化设计的结果对比，可以看到两者的结果几乎一致。

	C_d (count)		C_l		α
	ADflow	mSANN	ADflow	mSANN	
NACA0012	97.21	97.17	0.500	0.491	3.986
Opt_{ADflow}	92.17	92.36	0.499	0.499	1.863
Opt_{mSANN}	92.18	92.36	0.500	0.500	1.864

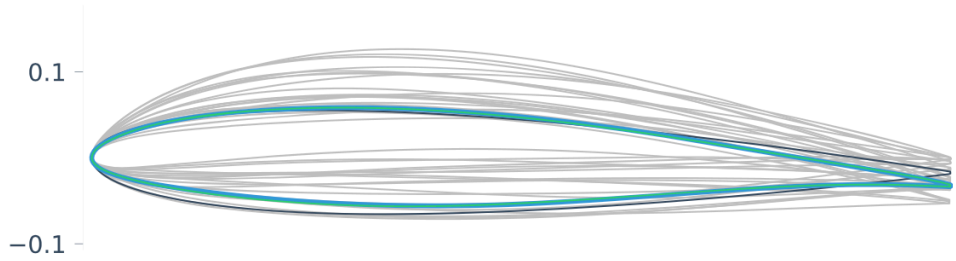


图 4.2 mSANN 和 ADflow 对 NACA0012 进行优化的结果对比[18]

4.2.2 跨音速翼型优化

对于跨音速翼型，本文选择 NACA SC(2)-0710 翼型作为测试算例。马赫数固定在 0.72，以 8 个模态系数和迎角作为设计变量，并对升力系数设置约束为$C_l = 0.82$，对阻力系数C_d

进行最小化。与亚音速类似，为了避免非正常翼型，将一阶厚度模态控制在 NACA SC(2)-0710 的一阶厚度模态的 90%以下，其余 7 个模态系数按照 3.2.1 节中的方法进行约束。

图 4.3 展示了 mSANN 方法和 CFD 方法进行优化设计的结果对比。由于流动状况的非线性情况更多，跨音速翼型的优化设计与亚音速相比会复杂许多，但两者的优化结果仍然几乎一致，表明基于 mSANN 的优化方法的有效性。

	C_d (count)		C_l		α
	ADflow	mSANN	ADflow	mSANN	
$SC(2)-0710$	114.44	1143.87	0.819	0.817	1.343
Opt_{ADflow}	101.31	101.36	0.820	0.818	0.848
Opt_{mSANN}	101.43	101.47	0.822	0.820	0.864

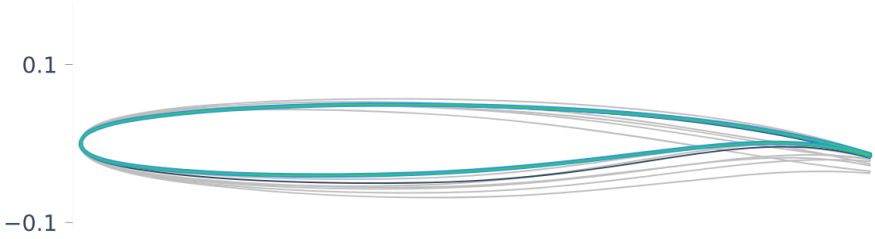


图 4.3 mSANN 和 ADflow 对 NACA SC(2)-0710 进行优化的结果对比[18]

4.2.3 翼型优化设计的效率

ADflow 进行 RANS 计算的时间与计算机硬件配置有关。实际测试中，在南京航空航天大学高性能计算平台 CPU2 队列（每个计算节点配备 2 个英特尔志强 8358CPU，每个 CPU 具有 32 核，节点内存为 256G）使用 64 核并行计算一个亚音速样本大约需要 120 秒，计算一个跨音速样本大约需要 50 秒。计算 2000 个样本，亚音速用时约 66.6 小时，跨音速用时约 28 小时。mSANN 的训练时长约为 1 小时左右（所用硬件为个人 MacBook Pro 笔记本）。直接使用 CFD 方法的翼型优化耗时在 20 到 60 分钟之间，而基于 mSANN 的翼型优化耗时只有几秒钟。考虑到训练时长，如果需要优化的翼型数量较少，那么基于 mSANN 的优化的总耗时比基于 CFD 的优化要多得多；但是当需要优化的数量超过 100 个时，耗时可以被平均，这时 mSANN 成为了更节约时间的方法。

第五章 总结与展望

5.1 研究总结

本研究探索了利用梯度强化人工神经网络作为代理模型对翼型的气动力参数进行预测的能力，并将其应用于翼型优化设计当中。改进的梯度强化人工神经网络 mSANN 通过在传统神经网络的损失函数中引入梯度以及权重系数λ_k，提高了神经网络的预测能力和收敛能力，并利用 Rosenbrock 函数测试验证了 mSANN 的性能要优于 ANN，SANN。为了获得训练神经网络所需的数据，使用奇异值分解，反距离加权插值方法，拉丁超立方采样等方法进行翼型数据预处理。选用开源 CFD 软件 ADflow 对样本翼型进行气动计算，并调用离散伴随方法计算气动力参数的梯度信息，建立了训练集和验证集。对神经网络进行训练，并耦合优化软件包 SNOPT 进行翼型优化设计。无论是亚音速还是跨音速条件，优化结果与基于 CFD 方法的优化设计结果几乎一致。建立了翼型数据的神经网络后，优化过程耗时约 4 秒，与基于 CFD 的优化相比大大降低了优化所需的时间。

5.2 研究展望

由于时间有限，很多方面的功能有所简化或不够完善。但是所实现的功能为开发翼型快速优化工具提供了基础框架，为了方便用户使用，后续可编写一套美观实用的用户界面。另外，本研究对于流动状态参数，只选择了迎角和马赫数作为训练输入，而雷诺数也是重要参数，可添加其中，适用于更大范围的翼型气动力预测和优化。此外，对于翼型模态系数的约束函数也可开展进一步的研究工作，探究如何设计约束方案既可以更好地避免奇怪翼型的出现，又能够保持尽量大的设计空间。

参考文献

[1] 方宝瑞．飞机气动布局设计[M]. 北京：航空工业出版社，1997.

[2] 白鹏，马汉东，周伟江. CFD 在大飞机设计中的工程化应用[C]. 中国航空学会 2007 年学术年会.

[3] 刘俊. 基于代理模型的高效气动优化设计方法及应用[D]. 西北工业大学，2017.

[4] 高正红，气动外形优化设计方法研究与存在的问题[C]，中国航空学会 2007 年学术年会.

[5] H. Gu, L. Yang, Z. Hu and J. Yu, "Surrogate Models for Shape Optimization of Underwater Glider"[C], 2009 International Conference on Computer Modeling and Simulation, Macau, China, 2009, pp. 3-6.

[6] J.-C. Jouhaud, P. Sagaut, M. Montagnac, J. Laurenceau. A surrogate model based multidisciplinary shape optimization method with application to a 2D subsonic airfoil [J]. Computers & Fluids 36 (2007) 520-529.

[7] Schmit, L. A., Farshi, B., Some approximation concepts for structural synthesis[J], AIAA Journal, Vol. 12, No.5, 1974, pp. 692-699.

[8] Zhang, K. S., Han, Z. H., Li, W. J., and Song, W. P., Bilevel adaptive weighted sum method for multidisciplinary multi-objective optimization[J], AIAA Journal, Vol. 46 No. 10, pp. 2611-2622. 2008.

[9] Xiong J. T., Qiao Z. D., Han Z. H., Aerodynamic shape optimization of transonic airfoil and wing using response surface methodology[C], Sep, 3-8, 2006, the 25th International Congress of Aeronautic Sciences, Hamburg, Germany.

[10] 张科施，韩忠华，李为吉．一种考虑气动弹性的运输机机翼多学科优化方法[J]．空气动力学学报，第 26 卷，第 1 期，2008.

[11] Vavalle, A., Qin, N., Iterative response surface based optimization scheme for transonic airfoil design[J], Journal of Aircraft, Vol. 44, No. 2, March-April 2007.

[12] Forrester, A. I. J., Keane, A. J., Recent advance in surrogate-based optimization[J], Progress in Aerospace Science, Vol.45, 2009, pp.50-79.

[13] Sobester, A., Leary, S. J., and Keane, A. J., On the design of optimization strategies based on global response surface approximation models[J]. Journal of Global Optimization, Vol. 33, pp. 31-59, 2005.

[14] Yun, Y., Yoon, M., and Nakayama, H., Multi-objective optimization based on meta-modeling by using support vector regression[J], Optimization and Engineering, Vol. 10, 2009.

[15] 穆雪峰, 姚卫星, 余雄庆, 刘克龙, 薛飞. 多学科设计优化中常用代理模型的研究[J]. 计算力学学报, 2005, (05): 608-612.

[16] X. He, J. Li, C.A. Mader, A. Yildirim, J.R.R.A. Martins, Robust aerodynamic shape optimization—from a circle to an airfoil, Aerosp. Sci. Technol. 87 (2019) 48–61.

[17] Li J , Du X , Martins J . Machine Learning in Aerodynamic Shape Optimization[J]. arXiv e-prints, 2022.

[18] Bouhlel, M.A., He, S. & Martins, J.R.R.A. Scalable gradient–enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes[J]. Structural and Multidisciplinary Optimization 61, 1363–1376 (2020).

[19] 张驰, 郭媛, 黎明. 人工神经网络模型发展及应用综述[J]. 计算机工程与应用, 2021, 57(11): 57-69.

[20] MCCULLOCH W S, PITTS W.A logical calculus of the ideas immanent in nervous activity[J]. Bulletin of Mathematical Biophysics, 1943,5: 115-133.

[21] 毛健, 赵红东, 姚婧婧. 人工神经网络的发展及应用[J]. 电子设计工程, 2011, 19(24): 62-65.

[22] HEBB D O. The organization of behavior: a neuropsychological theory[M]. New Jersey: Lawrence Erlbaum Associates, 1949.

[23] HOPFIELD J J. Neurons with graded response have collective computational properties like those of two-state neurons[J]. Proc Natl Acad Sci, 1984,81(10): 3088-3092.

[24] RUMELHART D E, HINTON G, WILLIAMS R J. Learning representations by back-propagating errors[J]. Nature, 1986,323(6088): 533-536.

[25] LECUN Y, BOTTOU L, BENGIO Yet al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998,86(11): 2278-2324.

[26] Angelova A, Krizhevsky A, Vanhoucke V (2015) Pedestrian detection with a large-field-of-view deep network[C]. In: Proceedings of ICRA 2015

[27] Ba J, Mnih V, Kavukcuoglu K (2014) Multiple object recognition with visual attention. arXiv:1412.7755

[28] Heigold G, Vanhoucke V, Senior A, Nguyen P, Ranzato M, Devin M, Dean J (2013) Multilingual acoustic models using distributed deep neural networks[C]. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Vancouver, CA.

[29] Rai MM, Madavan NK (2000) Aerodynamic design using neural networks[J]. AIAA J 38(1):173–182.

[30] Chen X, Agarwal R. Optimization of flatback airfoils for wind turbine blades using a genetic algorithm with an artificial neural network. In: 48th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition, 2010. p. 1423.

[31]Alberto Pliego Marugán, Fausto Pedro García Márquez, Jesus María Pinar Perez, Diego Ruiz-Hernández, A survey of artificial neural network in wind energy systems[J], Applied Energy, Volume 228, 2018, Pages 1822-1836, ISSN 0306-2619

[32] 朱国俊, 冯建军, 郭鹏程, 罗兴锜. 基于径向基神经网络-遗传算法的海流能水轮机叶片翼型优化[J]. 农业工程学报, 2014, 30(08): 65-73.

[33] 张玄武. 基于级联前向神经网络的翼型优化算法研究[D]. 浙江大学, 2017.

[34] 陈晨铭, 郭雪岩, 常林森. 基于 BP 神经网络代理模型的翼型优化及领域自适应研究[J]. 动力工程学报,2022, 42(07): 657-663.

[35] Giannakoglou K, Papadimitriou D, Kampolis I (2006) Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels[J], Computer Methods in Applied Mechanics and Engineering 195(44):6312–6329. ISSN 0045-7825

[36] Liu W, Batill S (2019) Gradient-enhanced neural network response surface approximations. In: 8th Symposium on Multidisciplinary Analysis and Optimization, Multidisciplinary Analysis Optimization Conferences AIAA.

[37] Pan S, Duraisamy K (2018) Long-time predictive modeling of nonlinear dynamical systems using neural networks. Complexity 2018:1–26.

[38] Czarnecki WM, Osindero S, Jaderberg M, Swirszcz G, Pascanu R (2017) Sobolev training for neural networks. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, pp 4281–4290.

[39] Li J, Bouhlel MA, Martins JRRA (2019) Data-based approach for fast airfoil analysis and optimization. AIAA J 57(2):581–596.

[40] 吴航空,王丁喜,黄秀全等.“定黏假设”对伴随系统求解和梯度精度影响[J].航空学报,2022,43(07):212-228.

学位研究期间取得的主要成果

一、 已获专利或软件著作权

软件著作权：

名称：无人车编队导航及队形切换系统；著作权人：南京航空航天大学；登记号：2022SR0384726；本人排名：2

二、 曾获相关学科竞赛成绩

1) I 级乙等，RoboMaster2020 机甲大师对抗赛(线上)，长空御风战队，二等奖，2020/08.
2) I 级乙等，中国机器人大赛暨 ROBOCUP 机器人世界杯中国赛(医疗机器人项目)，彭施聪，池烨恒，陈和灏，李鑫鹏，李晓童，三等奖，2020/11.
3) Ⅱ级乙等，第六届江苏省大学生工程训练综合能力竞赛飞行器设计与仿真，李晓童，王威奇，李鑫鹏，陈欣，二等奖，2021/04
4) Ⅱ级乙等，RoboMaster2021 机甲大师超级对抗赛(中部赛区)，长空御风战队，二等奖，2021/08.
5) I 级乙等，RoboMaster2021 机甲大师超级对抗赛(全国赛)，长空御风战队，三等奖，2021/11.
6) I 级乙等，中国机器人大赛暨 ROBOCUP 机器人世界杯中国赛(3D 识别)，左清宇，陈宏昱，李晓童，李博文，王威奇，三等奖，2022/04.
7) I 级乙等，中国机器人大赛暨 ROBOCUP 机器人世界杯中国赛(工业测量)，左清宇，王威奇，李博文，李晓童，陈宏昱，三等奖，2022/04.

三、 曾主持或参加的大学生创新创业训练计划

8) 省级，微小型固定翼攻击无人机系统，刘润甫，2020/04 至 2021/04.
9) 校级，物流分拣机器人定位及识别方法研究，陈宏昱，2021/04 至 2022/04.
10) 国家级，无人车动态队形变换与避障，李晓童，2021/04 至 2022/04.

附录 A prefoil 翼型预处理（统一化）代码

```
from prefoil import Airfoil, sampling

from prefoil.utils import readCoordFile

import numpy as np

import os

import math

import matplotlib.pyplot as plt


# read file list

sublib = 'coord_seligFmt'

subobj = '/Users/lxt/Desktop/毕设/code/data/sub'


translib = 'sc2'

transobj = '/Users/lxt/Desktop/毕设/code/data/trans'


lib=sublib

obj=subobj


files = os.listdir(lib)

files_exist = os.listdir(obj+'/foil')

files_abnormal = os.listdir('/Users/lxt/Desktop/毕设/code/data/abnormal')


#skip .DS_Store

#files.remove('.DS_Store')


for f in files:


    if f not in files_exist and f not in files_abnormal:
```

```
#skip .DS_Store
if f != '.DS_Store':
    print(f)

    # the coordinate file name
    filename = lib+'/'+f
    coords = readCoordFile(filename,1)

    # create an instance of an airfoil
    airfoil = Airfoil(coords)

    # Sets the twist to zero, the chord to one, and the leading edge location to the origin
    airfoil.normalizeAirfoil(True,True,True)

    # number of sampling points
    n = 126
    #n = 151

    # coordinates for the upper surface
    coord_upper = np.zeros((n,2))

    # coordinates for the lower surface
    coord_lower = np.zeros((n,2))

    #container for camber and thickness
    camber = np.zeros((n,1))
    thickness = np.zeros((n,1))

    # sampling method
    pt = sampling.cosine(start=0.0, end=1.0, n=n, m=math.pi)
```

```
np.savetxt(obj+"/pointx.dat",pt)

#coords = airfoil.getSampledPts(251, spacingFunc=sampling.conical, func_args={"m":math.pi,
"coeff": 1})
coords = airfoil.getSampledPts(300, spacingFunc=sampling.conical, func_args={"m":math.pi,
"coeff": 1})

# find y coordinate of the upper surface for each sample position
for i in range(n):
    tempx, s = airfoil.findPt(pt[i], axis=0, s_0=0.4)
    coord_upper[i,0] = pt[i]
    coord_upper[i,1] = tempx[1]
    #if coord_upper[i,1] < 0:
      #    coord_upper[i,1] = coords[n-1-i,1]

# reset the coordinate so that the leading edge is at the origin
coord_upper[0,1] = 0.0

# find y coordinate of the lower surface for each sample position
for i in range(n):
    tempx, s = airfoil.findPt(pt[i], axis=0, s_0=0.6)
    coord_lower[i,0] = pt[i]
    coord_lower[i,1] = tempx[1]
    #if coord_lower[i,1] > 0:
       # coord_lower[i,1] = coords[n-1+i,1]

#np.savetxt("/Users/lxt/Desktop/毕设/code/data/lower.dat",coord_lower)

# reset the coordinate so that the leading edge is at the origin
coord_lower[0,1] = 0.0
```

```
#compute camber and thickness
for i in range(n):
    camber[i] = (coord_upper[i,1]+coord_lower[i,1])/2
    thickness[i] = coord_upper[i,1]-coord_lower[i,1]

#find the max value of camber
cambermax = max(camber)

#smoothing y coordinates
yupper0 = np.copy(coord_upper)
ylower0 = np.copy(coord_lower)
yupperk = np.copy(coord_upper)
ylowerk = np.copy(coord_lower)

re1 = 0
times=0
while re1 <= 0.003:
    times = times+1
    for i in range(1,n-1):
        if(cambermax != 0):
            p = math.sin(camber[i]/cambermax*math.pi/2)*0.2+0.05
        else:
            p=0.07
        yupperk[i,1] = p*(coord_upper[i-1,1]+coord_upper[i+1,1])+(1-p)*coord_upper[i,1]
    re1 = np.linalg.norm(yupperk-yupper0)/np.linalg.norm(yupper0)
    coord_upper=yupperk
print(times)
```

```
re2 = 0
times=0
while re2 <= 0.003:
    times = times+1
    for i in range(1,n-1):
        if(cambermax != 0):
            p = math.sin(camber[i]/cambermax*math.pi/2)*0.2+0.05
        else:
            p=0.07
        ylowerk[i,1] = p*(coord_lower[i-1,1]+coord_lower[i+1,1])+(1-p)*coord_lower[i,1]
    re2 = np.linalg.norm(ylowerk-ylower0)/np.linalg.norm(ylower0)
    coord_lower=ylowerk
print(times)

#plot foil
#fig1=airfoil.plot()
#plt.show()
plt.plot(coord_upper[:,0],coord_upper[:,1])
plt.plot(coord_lower[:,0],coord_lower[:,1])
plt.ylim(-0.4,0.5)
plt.show()

#compute camber and thickness
for i in range(n):
    camber[i] = (coord_upper[i,1]+coord_lower[i,1])/2
    thickness[i] = coord_upper[i,1]-coord_lower[i,1]

cmd = input("if the airfoil meets requirement?")

if (cmd != "n"):
```

```
        # output file name
        savefoil = obj+'/foil/'+f
        savecamber = obj+'/camber/'+f
        savethickness = obj+'/thickness/'+f

        # write the coordinates of the upper surface (from TE to LE)
        np.savetxt(savefoil,np.flip(coord_upper, 0))

        # write the coordinates of the lower surface (from LE to TE)
        with open(savefoil, 'a') as datfile:
        # skip the origin
            np.savetxt(datfile, coord_lower[1:])

        #save camber and thickness
        np.savetxt(savecamber,camber)
        np.savetxt(savethickness,thickness)
        print('saved')

    else:
        savefoil2 = "/Users/lxt/Desktop/毕设/code/data/abnormal/"+f
        # write the coordinates of the upper surface (from TE to LE)
        np.savetxt(savefoil2,np.flip(coord_upper, 0))

        # write the coordinates of the lower surface (from LE to TE)
        with open(savefoil2, 'a') as datfile2:
        # skip the origin
            np.savetxt(datfile2, coord_lower[1:])
        print('abnormal')
```

附录 B 建立翼型矩阵代码

```
import numpy as np
import os
import matplotlib.pyplot as plt

sub = '/Users/lxt/Desktop/毕设/code/data/sub'
trans = '/Users/lxt/Desktop/毕设/code/data/trans'

regime = sub

# read file list
files=os.listdir(regime+'/camber')

if regime == sub:
    #skip .DS_Store
    files.remove('.DS_Store')
    camber = np.zeros((126,1389))
    thickness = np.zeros((126,1389))
    foil = np.zeros((251,1389))
else:
    camber = np.zeros((151,21))
    thickness = np.zeros((151,21))
    foil = np.zeros((301,21))

#sort names
files.sort()

i=0
```

```
for f in files:

    # read file
    data1=np.loadtxt(regime+'/camber/'+f)
    data2=np.loadtxt(regime+'/thickness/'+f)
    data3=np.loadtxt(regime+'/foil/'+f)

    #data1[0] = 0
    #data2[0] = 0

    #append to container
    camber[:,i]=data1
    thickness[:,i]=data2
    foil[:,i]=data3[:,1]

    i=i+1

#save camber
np.savetxt(regime+'/camberMatrix.dat',camber)

#save thickness
np.savetxt(regime+'/thicknessMatrix.dat',thickness)

#save foil
np.savetxt(regime+'/airfoil.dat',foil)

#load x coord
x=np.loadtxt(regime+'/pointx.dat')

#plot c, t, and foil
```

```
upperfoil=camber[:,0]+thickness[:,0]/2

lowerfoil=camber[:,0]-thickness[:,0]/2

plt.plot(x,camber[:,0],color='orange')

plt.plot(x,thickness[:,0],color='royalblue')

plt.plot(x,upperfoil,color='r')

plt.plot(x,lowerfoil,color='r')

plt.ylim(-0.2,0.3)

plt.show()
```

附录 C 奇异值分解代码

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import interpolate

sub = '/Users/lxt/Desktop/毕设/code/data/sub'
trans = '/Users/lxt/Desktop/毕设/code/data/trans'

regime = sub

#svd function
def svd(M):
    """
    Args:
        M: numpy matrix of shape (m, n)
    Returns:
        u: numpy array of shape (m, m).
        s: numpy array of shape (k).
        v: numpy array of shape (n, n).
    """
    u,s,v = np.linalg.svd(M)
    return u, s, v

#load camber matrix
data1 = np.loadtxt(regime+'/camberMatrix.dat')

#load thickness matrix
data2 = np.loadtxt(regime+'/thicknessMatrix.dat')
```

```
#load airfoil matrix
data3 = np.loadtxt(regime+'/airfoil.dat')

#load x coord
x = np.loadtxt(regime+'/pointx.dat')
xnew = np.linspace(0,1,500)

#do svd
u1,s1,v1 = svd(data1)
u2,s2,v2 = svd(data2)
u3,s3,v3 = svd(data3)

#save modes
np.savetxt(regime+'/cambermode.dat',u1)
np.savetxt(regime+'/thicknessmode.dat',u2)

n1 = 0
n2 = 0

#expand s to diagonal matrix
if regime == sub:
    mat1 = np.zeros((126,1389))
    mat2 = np.zeros((126,1389))
    n1 = 126
    n2 = 7
else:
    mat1 = np.zeros((151,21))
    mat2 = np.zeros((151,21))
    n1 = 151
    n2 = 4
```

```
for i in range(len(s1)):
    mat1[i,i] = s1[i]
    mat2[i,i] = s2[i]

#save mode coefficients
np.savetxt(regime+'/cambermodecoef.dat',np.dot(mat1,v1))
np.savetxt(regime+'/thicknessmodecoef.dat',np.dot(mat2,v2))

#choose the number of modes
'''for i in range(n1-n2):
    u1[:,i+n2]=0
    u2[:,i+n2]=0'''

#mode sequence
i = 0

#restore camber and thickness
c1=np.dot(u1,np.dot(mat1,v1))
c2=np.dot(u2,np.dot(mat2,v2))
'''f1=interpolate.interp1d(x[:],c1[:,i],kind='cubic')
f2=interpolate.interp1d(x[:],c2[:,i],kind='cubic')
curve1=f1(xnew)
curve2=f2(xnew)'''

#restore foil
upperfoil=c1[:,i]+c2[:,i]/2
lowerfoil=c1[:,i]-c2[:,i]/2

xb = 0
yb = -0.4
```

```
for i in range(20):
    #draw camber, thickness, and foil
    plt.plot(x+xb,u3[:n1,i]+yb,color='royalblue')
    plt.plot(x+xb,u3[n1-1:,i]+yb,color='royalblue')
    #plt.plot(x+xb,u2[:n1,i],color='royalblue')
    #plt.plot(xnew,curve2,color='royalblue')
    #plt.plot(x,upperfoil,color='r')
    #plt.plot(x,lowerfoil,color='r')
    #plt.ylim(-0.2,0.3)
    xb = xb+1.3
    if i%5 == 4:
        xb = 0
        yb = yb-0.5
plt.show()
```

附录 D 拉丁超立方采样、模态系数约束以及 IDW 代码

```
import numpy as np
from pyDOE import lhs
import matplotlib.pyplot as plt
from scipy import interpolate
from simple_idw import simple_idw

sub = '/Users/lxt/Desktop/毕设/code/data/sub'
trans = '/Users/lxt/Desktop/毕设/code/data/trans'

regime = sub

n = 0
m = 21

if regime == sub:
    n = 7
    m = 1389
    h = 126
else:
    n = 4
    m = 21
    h = 151

#bound of coefficients
ub = np.zeros((4,1))
lb = np.zeros((4,1))
modecoef = np.zeros((2*n,m))
modes = np.zeros((h,2*n))
```

```
#read coefficients
modecoef[:n,] = np.loadtxt(regime+'/cambermodecoef.dat')[:n,]
modecoef[n:,] = np.loadtxt(regime+'/thicknessmodecoef.dat')[:n,]
samplecoef = np.zeros((2*n+2,2000))

#read modes
modes[:,:n] = np.loadtxt(regime+'/cambermode.dat')[:,:n]
modes[:,n:] = np.loadtxt(regime+'/thicknessmode.dat')[:,:n]

#modes coefficients bounds
f = []
ac = modecoef[:n,]
at = modecoef[n:-2,]
x = np.linspace(min(ac[0]),max(ac[0]),50)
y = np.linspace(min(at[0]),max(at[0]),20)
xx, yy = np.meshgrid(x, y)
xx = xx.flatten()
yy = yy.flatten()
for i in range(2*n):
    if i == 0 or i == n:
        f.append(None)
        continue
    z = simple_idw(ac[0], at[0], modecoef[i,], xx, yy)
    f.append(interpolate.interp2d(x, y, z, kind='cubic'))
  # print(f[i])

#angle and Mach number bound
ub[0] = max(modecoef[0,])
lb[0] = min(modecoef[0,])
```

```
ub[1] = max(modecoef[n,])
lb[1] = min(modecoef[n,])
ub[2] = 4.5
lb[2] = -1.5
ub[3] = 0.85
lb[3] = 0.7


#lhs sample
samplecoef[0] = lb[0]+(ub[0]-lb[0])*lhs(1,2000).transpose()
samplecoef[n] = lb[1]+(ub[1]-lb[1])*lhs(1,2000).transpose()
samplecoef[-2] = lb[2]+(ub[2]-lb[2])*lhs(1,2000).transpose()
samplecoef[-1] = lb[3]+(ub[3]-lb[3])*lhs(1,2000).transpose()
ac1 = samplecoef[0]
at1 = samplecoef[n]
for i in range(2*n):
    if i == 0 or i == n:
        continue
    for j in range(2000):
        # print(i)
        lowerbd = min(0,f[i](ac1[j],at1[j]))
        upbd = max(0,f[i](ac1[j],at1[j]))
        samplecoef[i,j] = lowerbd+(upbd-lowerbd)*lhs(1,1).transpose()



#restore c,t,and foil
camber = np.dot(modes[:,:n],samplecoef[:n,:])
thickness = np.dot(modes[:,n:],samplecoef[n:2*n,:])
upperfoil = camber+thickness/2
lowerfoil = camber-thickness/2
upperfoil[0,:] = 0
```

```
lowerfoil[0,:] = 0


#load x coord
x = np.loadtxt(regime+'/pointx.dat')


print(samplecoef.shape)


plt.plot(x,upperfoil[:,:1],color='orange')
plt.plot(x,lowerfoil[:,:1],color='royalblue')
plt.ylim(-0.2,0.3)
plt.show()


#save camber
np.savetxt(regime+'/foilsample.dat',np.flip(upperfoil,0))
with open(regime+'/foilsample.dat', 'a') as datfile:
                    # skip the origin
                    np.savetxt(datfile, lowerfoil[1:])


#save states
np.savetxt(regime+'/statesample.dat',samplecoef[-2:,:])


y = np.vstack((np.flip(upperfoil,0),lowerfoil[1:]))
x = samplecoef[:8,:]
dy = np.zeros((2*n,2*h-1,1999))
for i in range(2*n):
    for j in range(2*h-1):
        dy[i,j,:] = np.diff(y[j,:])/np.diff(x[i,:])


#save gradient
#np.savetxt(regime+'/gradient.dat',dy)
```

```
import numpy as np

def distance_matrix(x0, y0, x1, y1):
    obs = np.vstack((x0, y0)).T
    interp = np.vstack((x1, y1)).T

    # Make a distance matrix between pairwise observations
    # Note: from <http://stackoverflow.com/questions/1871536>
    # (Yay for ufuncs!)
    d0 = np.subtract.outer(obs[:,0], interp[:,0])
    d1 = np.subtract.outer(obs[:,1], interp[:,1])

    return np.hypot(d0, d1)

def simple_idw(x, y, z, xi, yi):
    dist = distance_matrix(x,y, xi,yi)

    # In IDW, weights are 1 / distance
    weights = 1.0 / dist

    # Make weights sum to one
    weights /= weights.sum(axis=0)

    # Multiply the weights for each interpolated point by all observed Z-values
    zi = np.dot(weights.T, z)
    return zi
```

致　　谢

本文的研究工作是在高宜胜老师的认真指导下完成的。高老师为人和善，学识渊博，治学严谨，不管在研究方法层面还是在具体编程实现方面都给予我以悉心指导。本次研究的工作量较大，在截止时间的最后高老师仍然在帮助我解决遇到的问题。如果不是高老师的督促和指导，我很难在如此短的时间内完成本文的所有工作。在对待每一次答辩和 PPT 制作时，高老师严谨认真的要求也让我深受影响，认识到自己在科研方面的态度还有待改善。除了对于毕业设计的指导，高老师对我有关未来学术生涯的困惑也进行了非常耐心的解答，向我分享他自身的经验，供我借鉴。在此向高老师致以衷心的感谢！

感谢父母对我的养育，这一路走来，他们一直陪伴着我，无论是喜悦还是烦恼，他们总会有耐心倾听我的想法，尊重我的选择。我能获得今天拥有的成绩，离不开他们开明的态度和对我的要求。另外，还要感谢我的舍友，我的朋友们，在南航四年里认识或者遇见的每一位给予过我帮助的老师与同学们，是你们让我的大学生活如此丰富，如此快乐，也如此难忘。

最后，感谢南航，永远的母校；感谢南京，这座对我来说最为亲切的城市。不久之后，我就要踏上更为遥远的求学之路，可以预见，在异国他乡，我会无数次地怀念起这四年里，在这片土地上我所看过的风景和我所遇见的人。感谢你们！

毕业论文是我四年本科求学生活的最后一块拼图，致谢是毕业论文的最后一个部分。

此刻我拼完了拼图。