第五章 健康管理系统数字仿真平台原型系统的开发和验证

5.1 引言

在 V 型系统工程开发流程中，工程设计开发和集成验证是最后两个阶段。在工程设计开发时，需要根据每条既定的具体需求并结合现有技术条件，首先确定系统软件架构并划分系统的各个功能模块，然后针对各功能设计相应算法和代码结构，最后形成完整具体的技术路线。在集成验证时，首先需要保证代码编写不出现错误，其次根据功能需求检查各个功能是否能正确运行，最后对测试整个系统的可靠性和稳定性。

符合系统设计流程的航空发动机健康管理系统数字仿真平台可减少真实系统以及其中关键技术的开发周期并降低开发难度和成本。此外为了对本文在第三、四章提出传感器故障诊断和剩余使用寿命方法进行进一步地集成验证，本章基于第二章对航空发动机健康管理数字仿真平台的需求分析和方案设计，综合考虑科研硬件条件设计开发该原型系统。该原型系统仍然采用机载、地面、用户终端三大健康管理子系统的架构设计。

本章首先介绍了原型系统的功能划分和软件架构；再介绍了原型系统所用到的工程开发技术；然后对三个子系统进行了详尽的工程设计，并验证了对应部署情况；最后在一个仿真实例中验证了原型系统的核心功能。

5.2 整体架构

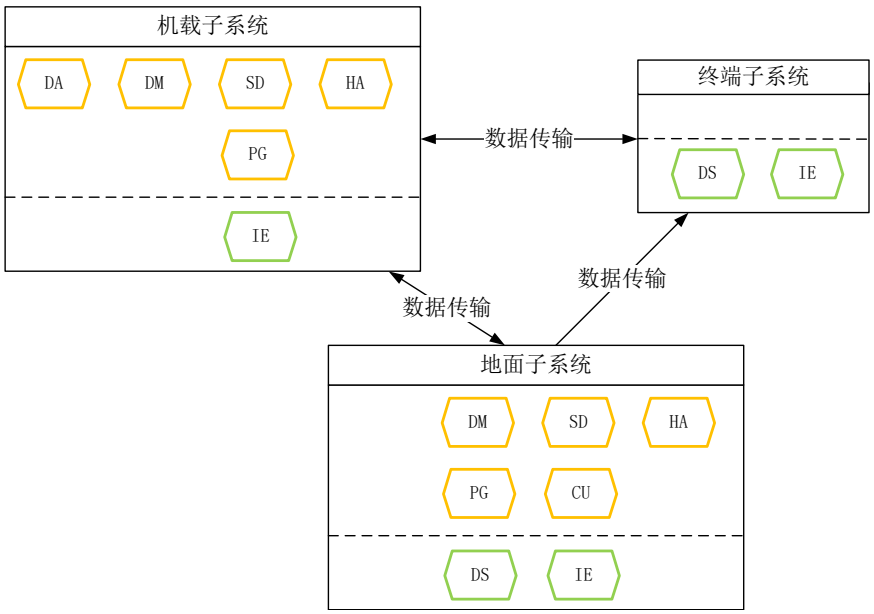


图 5. 1 健康管理系统仿真平台整体功能架构示意图

基于第二章对数字仿真平台的方案设计，仿真平台需要具备：数据采集（DA）、数据处理（DM）、状态检测（SD）、健康评估（HA）、建议生成（PG）、配置更新（CU）六大功能，以及数据存储（DS）、信息表征（IE）和数据传输等基础功能。数字仿真平台的功能架构如图 5.1 所示。

根据数字仿真平台的功能设定和功能架构，本文原型系统采取的整体架构如图 5.2 所示。如图 5.2 所示，机载健康管理子系统部署在具有仿真环境的普通计算机上，为了保证传输带宽和速率以及仿真稳定性，通过局域网与地面健康管理子系统连接。地面健康管理子系统部署在高性能 GPU 服务器上，拥有大量的存储资源和计算资源，同时地面子系统也和云端连接，通过云端给用户终端健康管理子系统发送运维信息，同时用户可通过云端对地面子系统进行访问。用户终端管理子系统的理想部署设备为移动设备，可以方便用户随时接受和查看运维信息。该架构符合典型的 C/S 架构，地面子系统为 C/S 架构中服务器为机载子系统和用户终端子系统两个客户端提供数据支持和计算资源等服务。

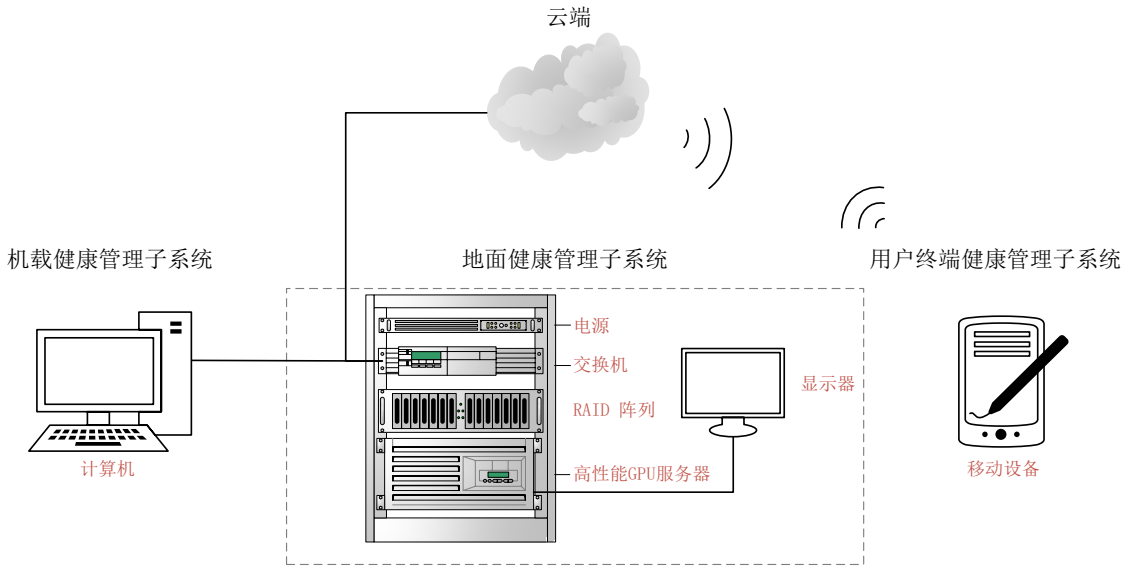


图 5.2 原型系统整体架构示意图

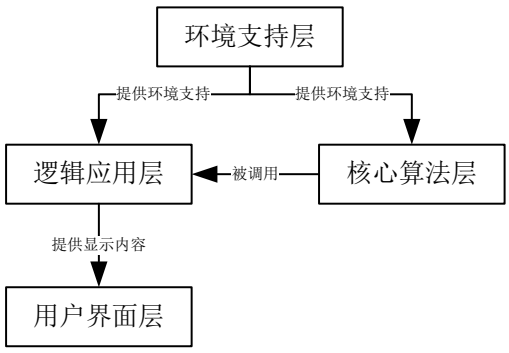


图 5.3 原型系统软件架构设计

基于第二章对每个子系统的方案设计，原型系统的子系统软件架构均由环境支持层、逻辑应用层、核心算法层、用户界面层组成，如图 5.3 所示。

（1）环境支持层：为原型系统各子系统提供基本的软硬件运行条件。

（2）逻辑应用层：逻辑应用层根据需求分析，定义原型系统的各应用或模块之间和之内的基本运行逻辑，包括功能配置、参数配置、权限配置等等。逻辑应用层通过调用核心算法层中的各类算法实现相关系统核心功能。

（3）核心算法层：根据需求定义的各原型系统核心功能，提供各功能所需算法被逻辑应用层调用。例如本原型系统的核心算法包括有：第三章和第四章所涉及的基于 Inception-CNN 的传感器故障诊断算法和基于 Resnet 和 LSTM 的双任务剩余使用寿命预测算法以及故障注入算法等。

（4）用户界面层：将用户操作和后台代码和运行逻辑隔离，保证系统稳定运行的同时提供人性化的操纵逻辑方便用户的使用。

5.3 主要工程开发技术介绍

● Python

在原型系统的工程开发中，Python 编程语言被大量使用，包括算法设计、逻辑应用、数据库操作、UI 设计等等。Python 是出色的面向对象、解释型和交互式的编程语言，它由荷兰数学家和计算机学家 Guido van Rossum 在 1990 年设计，起初 Python 主要用于编写自动化脚本，但随着语法的进一步精炼和更多功能强大库、模块的引入，Python 正在被更多大型项目所使用，例如 Zope、Mnet 以及 NASA、Google 等大型公司和组织都在广泛地使用它。此外 Python 在科学计算领域有着独一无二的优势，一方面有着 Pandas、NumPy、PyTorch 等第三方科学计算库使 Python 拥有丰富的数据格式并能够进行矩阵运算和非线性建模等等，另一方面 Python 在文件管理、界面设计、网络通信方面有着多样的扩展库，可以将科学计算融入至各种高级任务中。所以使用 Python 实现健康管理数字仿真平台原型系统是合适的。

● PyTorch

PyTorch 于 2017 年 1 月，由 Facebook 人工智能研究院（FAIR）推出，PyTorch 是开源基于 Python 的深度学习框架。PyTorch 的核心在于构建、优化和训练深度神经网络，为图像、语音、视频等大规模机器学习任务提供高效快速的计算方案。PyTorch 主要包括两个高级功能：1.能够利用 GPU 强大的并行运算能力进行张量计算。2. 搭建具有自动反向求导机制的动态深度神经网络。PyTorch 的设计非常简洁，遵循张量（tensor）→变量（variable）→神经网络模块（module）三个由低到高的抽象对象层次，三个抽象对象紧密联系而且可通过参数对属性进行修改，这给予用户足够的自由度，可以使用户更深度 API 理解底层代码。在本文中，提出的传感器故障诊

断和剩余使用寿命预测方法均由 PyTorch 实现。

● Django

Django 基于 Python 的开源 Web 框架。Django 的核心设计是 MTV 模式，即 Model（模型）、View（视图）、Template（模板）三个模块，具体由图 5.4 所示。

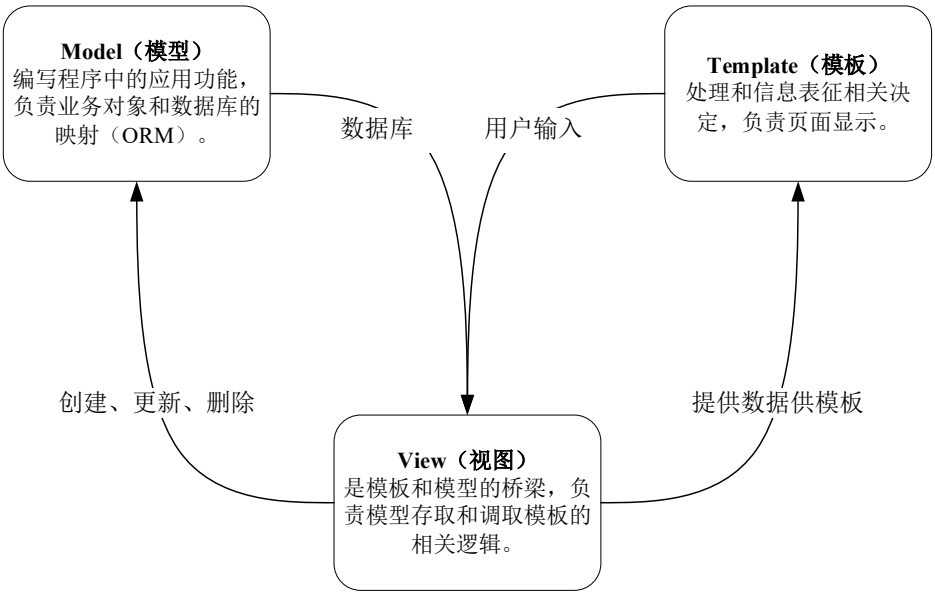


图 5. 4 Django 的 MTV 模式

除开图 5.4 所示的 3 个模块，Django 还有一个 URL 层，其作用是将 URL 页面请求分发给不同的 View 文件处理。得益于 MTV 架构，Django 在拥有强大数据库和后台功能的同时能够在较少代码情况下快速开发、设计网站。本文选用 Django 作为原型系统的实现的原因主要有 4 点：

（1）拥有强大数据库功能和 ORM 层，在不精通数据库语言情况下可以方便地实现模型库和算法库一起其他必要库。

（2）MTV 模式可以方便地将逻辑应用编写和 UI 界面设计相隔离。

（3）相对于 Python 的其他 GUI 设计模块，Django 可以依赖 Web 前端设计语言和框架，UI 界面可以设计得更加人性化。

（4）Django 是 Web 框架可以部署在服务器上，其他设备通过网络可直接访问，而不需要像传统应用程序一样进行打包分发安装。

● Unity

Unity 是 Unity 科技公司开发的实时三维动画、三维视频游戏、工程项目可视化、建筑实景仿真的综合性创作开发工具。Unity 的优势在于支持的平台广泛和易用性，Unity 拥有可视化编程开发界面，支持所见即所得，拥有出色的开发效率。

在计算机软硬件快速发展的今天，在不少工程领域都使用数字孪生技术缩短开发周期、降

低维护成本，包括本文的数字仿真平台也是一种数字孪生技术在航空发动机健康管理系统上的应用。数字孪生技术的一个重点就是搭建符合物理规律的虚拟模型，所以三维模型创建工具Unity在各行各业得到了广泛的应用。

在本章中，Unity被使用开发原型系统机制子系统的部分用户界面。

- 其他

本章除开使用上述4个主要技术开发原型系统外，还使用TCP/IP传输协议连接机载子系统和地面子系统，使用MQTT通讯协议连接机载子系统控制端和Unity开发的可视化三维模型端，使用Python科学计算包NumPy和Pandas进行数据操作和矩阵运算等，使用前端语言HTML和Javascript以及Bootstrap工具包开发前端网页等等。

5.4 机载健康管理子系统

5.4.1 工程设计开发

基于2.3节对仿真平台机载子系统的方案设计，原型系统的机载子系统在软件架构上有如下设计，如图5.5所示。从图5.3所示的环境应用层、逻辑应用层、核心算法层、用户界面层描述机载子系统软件结构。

（1）环境应用层：

为了保证较小的仿真步长和较快的仿真速率，在硬件方面，机载子系统需采用Intel® Core™ 8th Gen i5及以上性能CPU和16GB以上RAM以及分辨率在1920×1080以上彩色显示器。在软件方面，机载子系统需要运行在Windows或者具有图形界面的Linux系统上，本原型系统使用Windows 10为例。此外本文使用Python语言完成逻辑应用和核心算法的代码编写，所以在机载子系统需要具备Python语言运行环境，本原型系统使用Python版本为3.8。

（2）逻辑应用层：

该层主要由控制模块、仿真模块和监听模块三部分构成。该层主要由Django框架实现。

- 控制模块：控制模块是软件架构的主进程，主要负责系统的数据传输、数据库操作和子线程控制。在数据传输中，首先机载子系统建立和地面子系统的TCP/IP连接，建立逻辑应用层和用户界面层的Unity三维可视化模型端的MQTT连接。接着，地面子系统返回可访问的模型对象和算法对象名称，根据用户在机载子系统网页端选择的模型和算法将模型和算法对象下载至机载子系统的RAM中。然后用户可在控制模块中控制仿真启停，即创建和删除仿真子线程和监听子线程。此外在仿真过程中用户也可通过控制模块控制子线程的全局变量以达到控制仿真速度和注入传感器故障的目的。

- 仿真模块：仿真模块是机载子系统的核心，是检验健康管理系统关键技术的必要过程。首先，当用户启动仿真后，仿真模块创建仿真记录的DataFrame对象，用于记录算法输入和输

出。接着仿真模块创建仿真子线程，该子线程每次循环都按时刻读取模型对象，经过设定的间隔时间后，子线程会检查是否收到故障注入信号，若没有收到则直接将原始模型数据提交给传感器算法进行推理，而如果收到注入信号，则将原始模型数据经过故障注入算法进行修改，具体可注入的传感器故障如表 3.2 所示，再将修改后的数据提交算法推理。当模型读取迭代完成，子线程停止，仿真记录模块收集完整航程数据。RUL 预测算法如第四章所述，首先将仿真记录中 RUL 预测算法所需的完整航程数据提取并进行上采样或下采样统一数据长度，提交给 RUL 预测算法进行推理的出当前航程剩余寿命预测结果。最后，将仿真记录对象存入 RAM，并通过 Socket 连接传输至地面子系统进行长期存储。

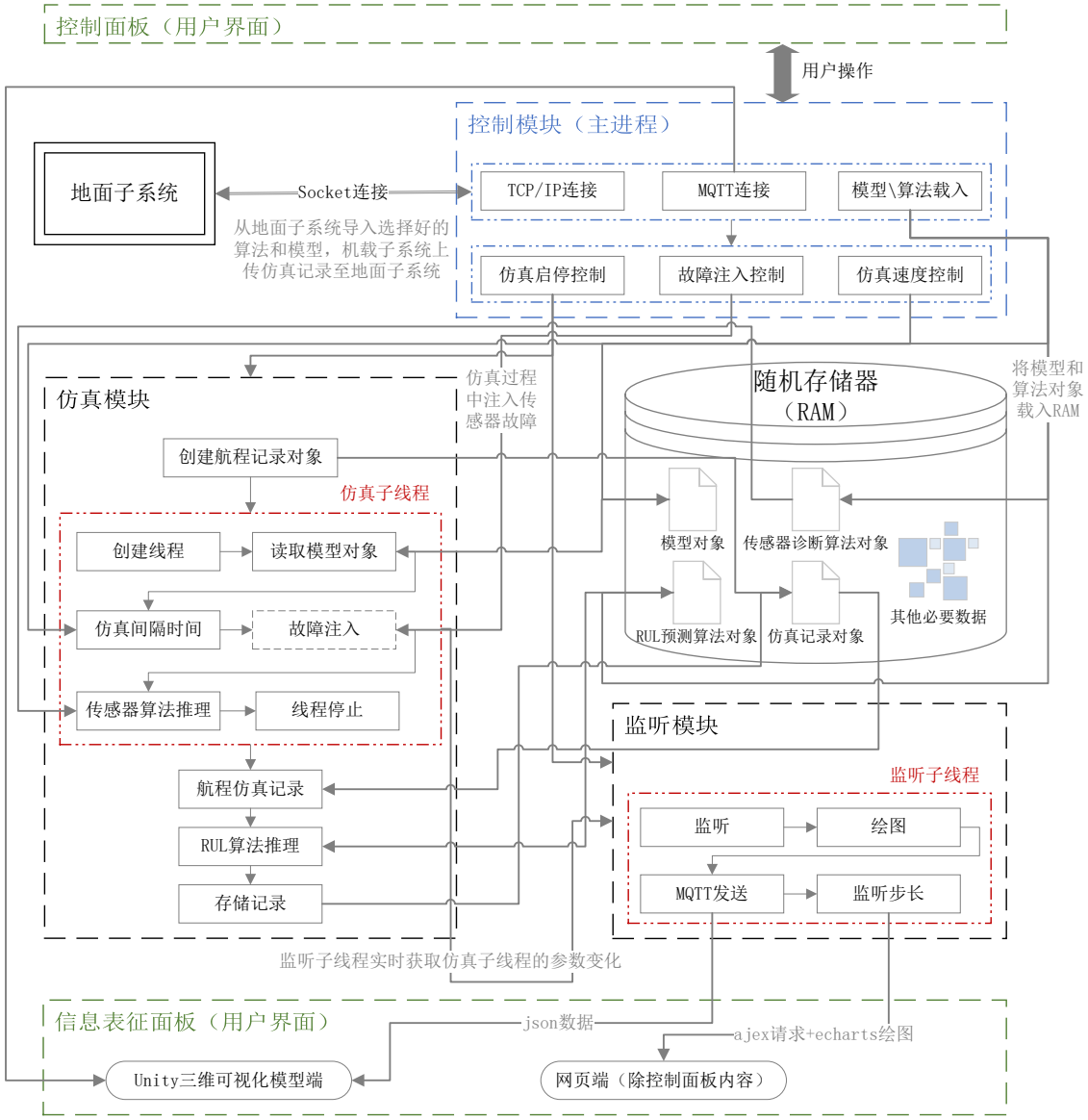


图 5.5 原型系统机载子系统软件架构设计图

- 监听模块：监听模块主要为用户界面更新服务。由于实时更新显示数据和三维动画对计

算机是不小的负担，所以需要通过监听模块的监听步长平衡显示和仿真之间的计算资源。当仿真子线程启动时，监听子线程也随之启动，监听子线程通过读取仿真子线程的全局变量获取数据变化，接着将获取的数据一方面在 Web 界面通过 echarts 和 ajex 请求绘制图表，另一方面将数据转换为 json 格式通过 MQTT 连接传输至 Unity 三维可视化模型端并用于图标和三维动画的更新。最后监听子线程将根据监听步长等待一段时间后再开启下一轮监听。

（3）核心算法层：

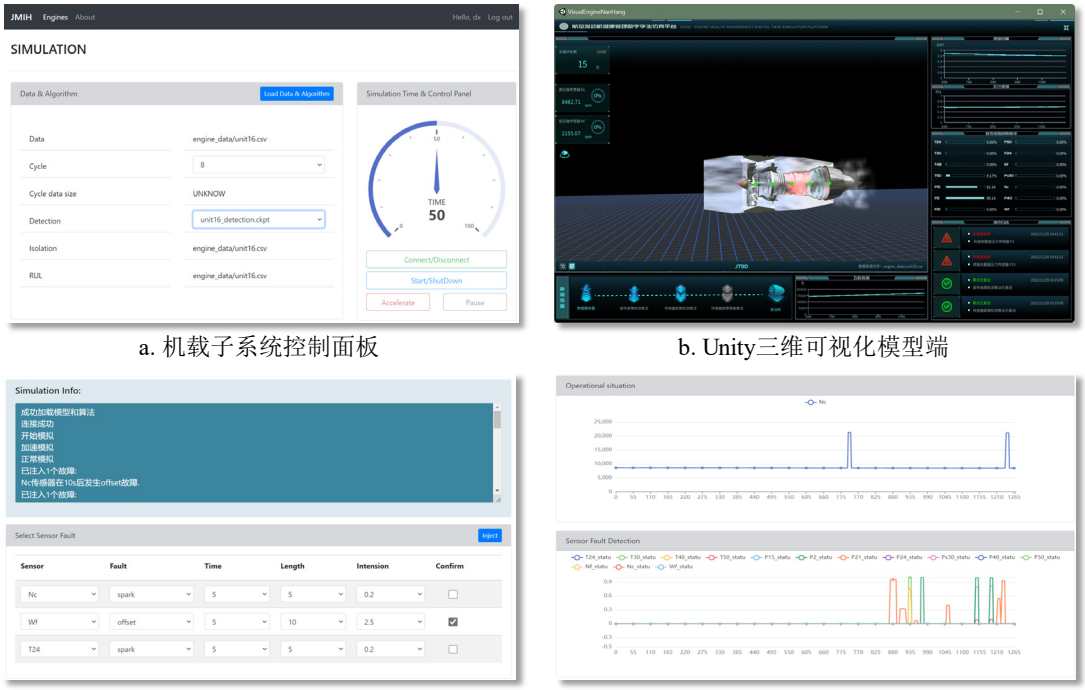
机载子系统通过用户在控制面板所选择的算法从地面子系统抓取对应算法对象。算法对象是带权重的 PyTorch 神经网络模型 nn.Module 对象。机载子系统在仿真子线程循环过程中调用传感器诊断算法，即第三章提出的基于 Inception-CNN 的传感器故障诊断算法，在仿真子线程结束后即获取完所有的航程数据后调用剩余使用寿命预测算法，即第四章提出的基于 Resnet 和 LSTM 的双任务剩余使用寿命预测算法。

（4）用户界面层：

机载子系统的用户界面层主要由 Web 网页端和三维可视化模型端构成。Web 网页端利用 Django 框架实现，Web 网页端包括机载子系统的控制面板和实时仿真数据显示。控制面板包括仿真启停、故障注入、数据连接等，实时仿真数据显示有操作信息反馈和算法输入前数据展示和推理后数据展示等。三维可视化模型端由 Unity 实现，三维可视化模型端提供了更详尽和直观的数据展示，其中的三维发动机模型会随着输入数据的不同有着不同的状态变化，而且在三维可视化模型端，可以方便地从三维模型中查看每个传感器的位置并查看其当前和历史数值。此外三维可视化端为算法推理结果也提供了直观的展示方式，用户方便地通过可视化段评估算法性能。

5.4.2 部署验证

图 5.6 是原型系统机载子系统的部署情况。如图 5.6 所示，a 图是控制面板，可以选择需仿真的模型和算法以及控制仿真启停和加速暂停等。b 图是 Unity 三维可视化模型端，可以更直观详细观察算法推理情况和传感器数值和位置等，图中 *P2* 和 *P15* 的传感器发生偏置故障并被算法诊断出。c 图是用户界面 Web 端的操作日志和故障注入面板，可以观察仿真模拟流程并检查仿真过程是否发生不可预知错误，仿真过程中可以最多可以同时注入三个不同的故障，并选择故障发生时间、长度、幅值、类型等。d 图是用户界面 Web 端算法推理结果的可视化，由监听模块将数据更新到 Web 端。



a. 机载子系统控制面板　　b. Unity三维可视化模型端

c. 操作日志和故障注入面板　　d. 用户界面Web端

图 5. 6 原型系统机载子系统部署情况

5.5 地面健康管理子系统

5.5.1 工程设计开发

基于 2.4 节对仿真平台地面子系统的方案设计，原型系统的地面子系统在软件架构上有如下设计，如图 5.7 所示。仍然从图 5.3 所示的环境应用层、逻辑应用层、核心算法层、用户界面层描述地面子系统软件结构。

（1）环境应用层：

为了保证地面子系统拥有足够的性能完成计算密集任务，在硬件方面，地面子系统需要在满足常规服务器硬件要求的前提下，搭载高性能和高显存的 GPU，本文推荐使用 Nvidia® Tesla® V100 以上性能的 GPU。在软件方面，地面子系统可运行在 Windows Server、Windows 或 Linux 上，此外需要安装 GPU 通用并行计算架构 CUDA™。地面子系统使用 Django 完成逻辑交互、数据管理、用户界面等应用，所以地面子系统需要具有 Django 运行环境，如 Python、JavaScript 语言环境等。

（2）逻辑应用层：

该层主要由用户管理、模型管理、算法管理、仿真配置和记录管理、终端子系统信息管理等五个模块构成，用户管理、模型管理、算法管理、仿真管理四个模块层层递进，下层模块是上层模块的子模块。

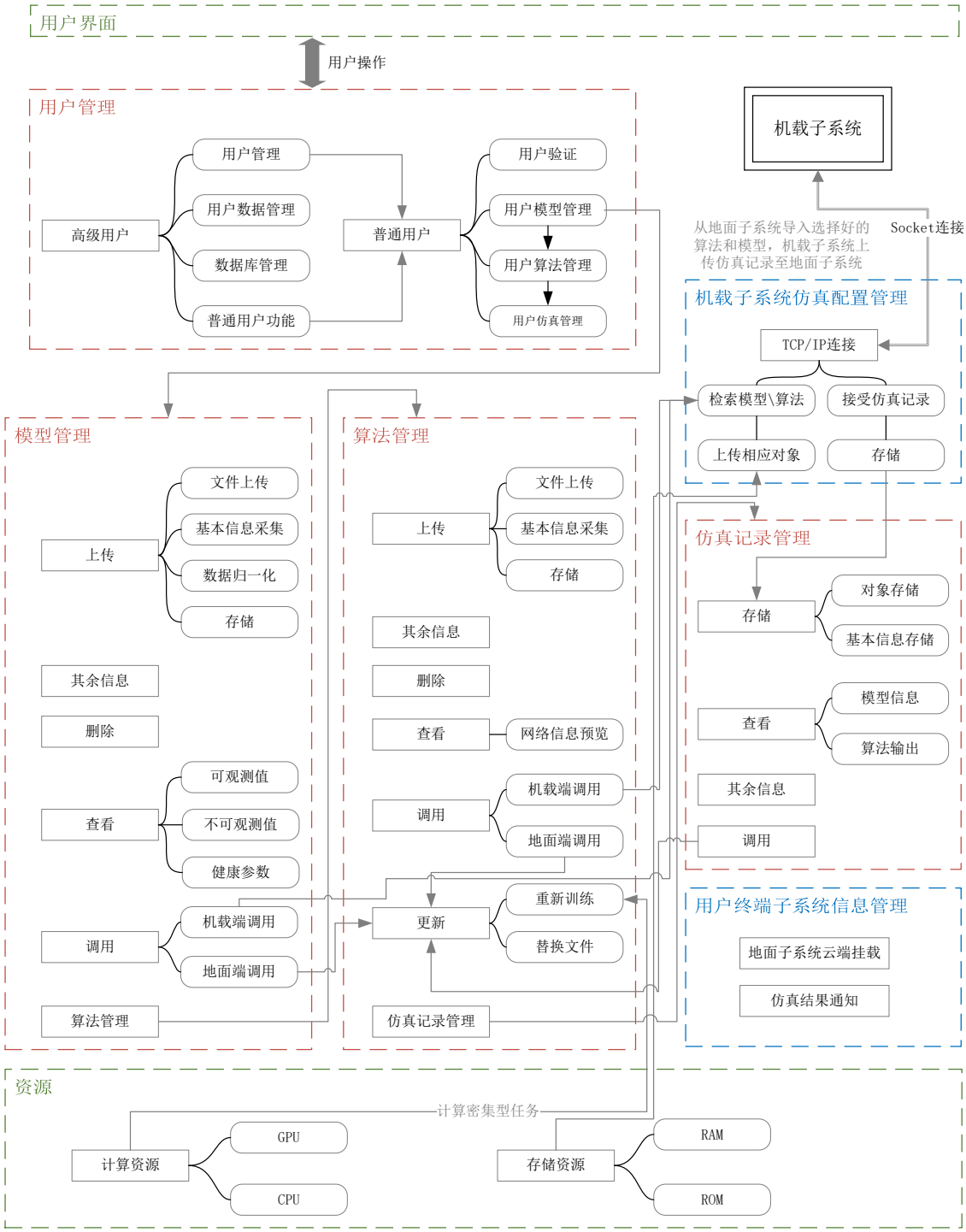


图 5.7 原型系统地面子系统软件架构设计图

● 用户管理：用户管理模块是地面子系统中优先级最高的管理模块，其余模块需要遵从用户管理模块规则。地面子系统利用用户管理模块将各用户的数据、操作等相隔离，这不仅提高用户信息安全性而且使地面子系统可以为多用户同时服务。在用户管理模块中，有超级用户和普通用户两个选项，超级用户可以对普通用户信息进行管理，可删除或新建普通用户，而且拥

有对地面子系统所有数据的操作权限。普通用户通过验证后，可使用该权限内的数据和执行操作。

● 模型管理：对应 2.4 节地面子系统设计，地面子系统需要有机群管理功能，此处原型系统通过模型管理模块实现该功能。在模型管理模块中，用户可创建航空发动机词条，并且在此词条下上传、删除航空发动机模型文件，每个词条可对应多个模型文件，即一个词条可看作相同发动机的机群。在上传模型文件后，系统根据既定规则提取该模型文件基本信息，如大小、时间长度等等，并且会自动将数据作归一化处理并存储以准备进一步作预览数据、算法推理等任务。用户可以为模型文件和词条额外添加信息，如实验记录、特性说明等等。模型数据可以提前预览，以供用户提前了解模型数据细节。模型可供机载端和地面端调用，机载端调用主要用作仿真模拟，地面端调用主要用作更新算法。在每个模型文件下，可查看和管理相应的算法。

● 算法管理：和模型管理相同，算法管理的一个词条下也可对应多个算法对象，比如在一个词条下可存储网络架构相同，但网络参数不相同的算法。同样，用户可为词条和算法文件添加额外信息。通过后台引入神经网络统计插件，用户可以预览算法网络架构的一些统计信息。在仿真模拟和算法更新时可在算法库中调用相应算法。在算法更新时去除原算法网络参数并利用模型数据、仿真模拟记录等数据重新训练神经网络。在每个算法文件下，可管理相应的仿真记录。

● 仿真管理：此处有仿真配置管理和仿真记录管理，地面子系统通过 TCP/IP 连接到机载子系统，一方面为机载子系统提供相应算法和模型上载，另一方面下载和存储仿真记录。在存储时，同时存储仿真时间，仿真时所使用模型和算法等相关信息。用户可为仿真记录添加额外信息。在更新算法时可调用仿真记录。

● 用户终端子系统信息管理：该模块决定了终端子系统可接受的信息和接受方式。该原型系统地面子系统提供了两种信息传递方式，一是将地面子系统挂载云端，用户可在移动设备上直接访问地面子系统；二是在机载子系统完成仿真模拟后，将仿真结果以邮件方式发送至用户终端。

（3）核心算法层：

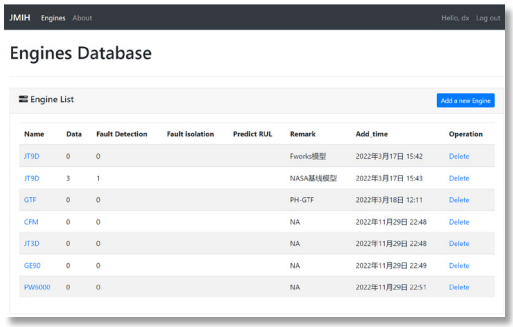
地面子系统通过算法管理模块管理传感器故障诊断算法和剩余使用寿命预测算法的 nn.Module 对象，通过仿真管理模块实现机载子系统对算法的调用。此外，地面子系统可通过仿真记录和模型数据以及原有算法对算法进行重新训练并更新。

（4）用户界面层：

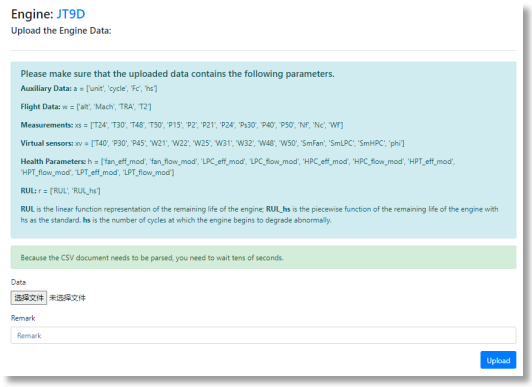
地面子系统的用户界面层由 Web 框架 Django 实现，用户界面与逻辑应用符合 Django 的 MTV 模式。页面中的数据图表部分使用 echarts 工具和 ajex 技术实现，UI 样式使用 Bootstrap 工具实现。

5.5.2 部署验证

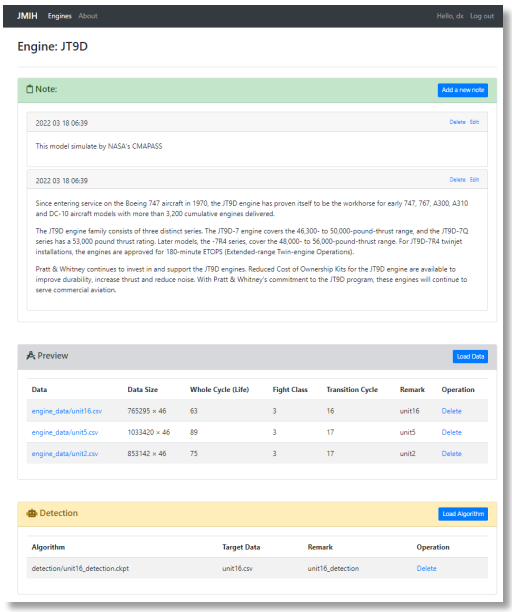
a.地面子系统首页



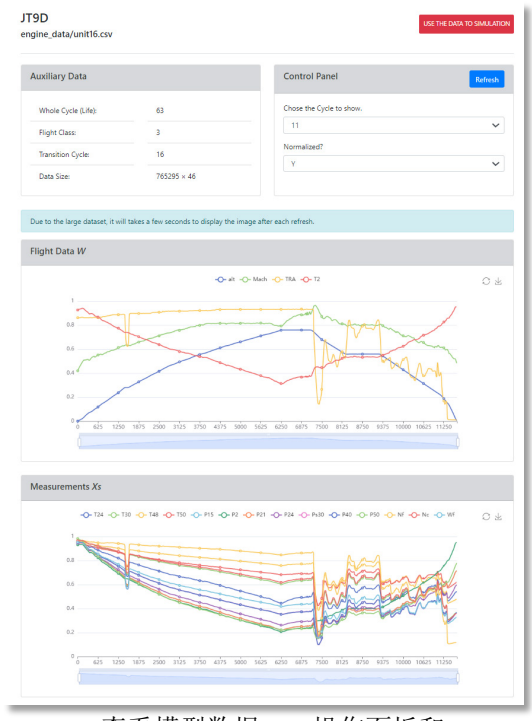
b.航空发动机库



c.上传模型文件



d.模型管理页面



e.查看模型数据——操作面板和可观测值

f.查看模型数据——不可观测值和健康参数

图 5. 8 原型系统地面子系统部分部署情况

图 5.8 是原型系统地面子系统的部分的部署情况。a 图是地面子系统的首页，简要介绍了系统功能。b 图是航空发动机库，发动机库中可以查看该用户所拥有的所有发动机，在此页面可以简要查看发动机信息。c 图是上传模型页面，在此页面给出了系统对模型格式的要求，并且用户在选择模型后可对其添加简短描述。d 图是发动机模型管理页面，在此页面用户可查看该发动机的详细信息，如：对应的算法、所包含的模型、发动机描述等等，算法管理页面和仿真记录页面和该页面类似。e 图和 f 图是查看模型数据页面，该页面可查看原始模型数据和归一化后的模型数据，了解模型数据在航程内的变化趋势，该页面可查看飞行数据、传感器值等可观测量，也可以查看模型虚拟传感器和健康参数这类的不可观测量，其中健康参数的横轴是该发动机的整个生命周期。

5.6 用户终端健康管理子系统

原型系统的用户终端子系统比较简单。从上一节可知，其主要通过移动设备云端访问地面子系统和接受仿真结果邮件构成，此处将不再赘述。

5.7 仿真实例验证

在此部分使用 N-CMAPSS 数据集中 DS02 数据集组的 16 号数据单元的模拟数据对原型系统的核心功能进行仿真实例验证。在数据集中不同的数据单元指代不同的航空发动机全生命周期模拟数据，该 16 号发动机具有 63 个从未知状态到部件发生故障的航程数据，此处选择该发动机第 40 次飞行的航程数据进行功能验证，所以该航程的真实剩余使用寿命为 23 次飞行航程。

通过故障注入功能，给正在运行的仿真模型注入 3 个将同时发生的传感器故障，分别是：*T30* 传感器在 5 秒后发生 10 秒幅值为原始数据 2.1 倍的漂移故障；*T50* 传感器在 6 秒后发生 12 秒幅值为原始数据 0.2 倍的偏置故障；*Nf* 传感器在 7 秒后发生 5 秒幅值为原始数据 2.5 倍的冲击故障。在原型系统故障注入模块的表示如图 5.9 所示。

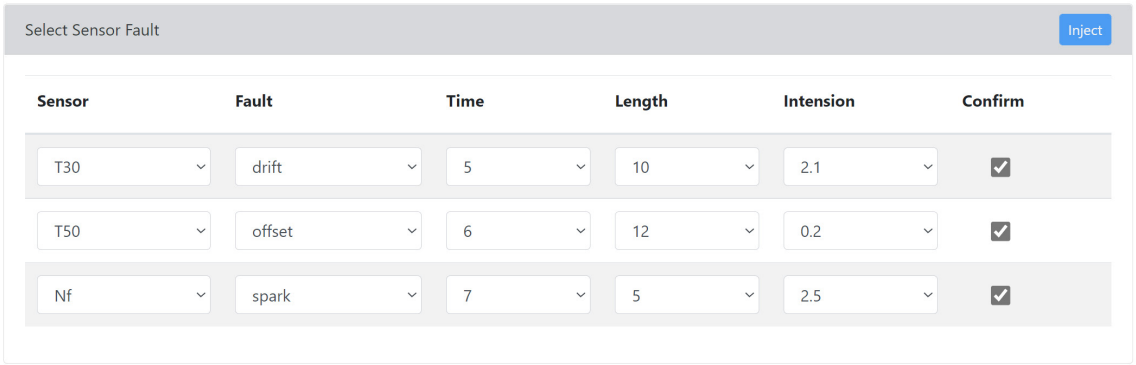


图 5. 9 仿真实例故障注入

根据第三章所述的传感器故障诊断方法，针对该仿真实例搭建 Inception-CNN 传感器故障诊断算法并集成至该原型系统，原型系统的机载子系统调用传感器故障诊断算法对模拟数据进

行实时监测和诊断。诊断结果如图 5.10 和 5.11 所示。

图 5. 10 仿真实例传感器故障诊断结果（Web 端）

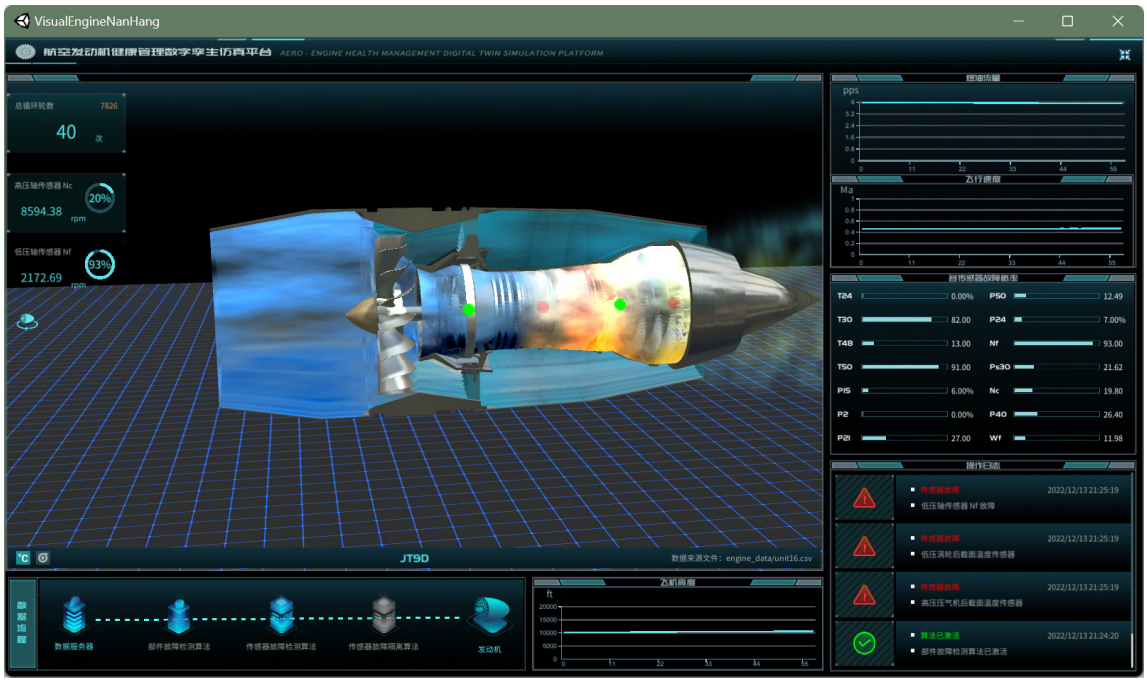


图 5. 11 仿真实例传感器故障诊断结果（Unity 三维可视化端）

图 5.10 和图 5.11 分别是该仿真实例在 Web 端和 Unity 三维可视化端的传感器故障诊断结

果，在 Web 可视化界面，图 5.10 的上半部分时注入故障传感器的历史数据，下半部分是全部传感器的历史故障概率。在 Unity 三维可视化端，用户可以通过三维模型更直观地了解航空发动机运行情况以及查看更多详细信息，如传感器位置，各传感器历史数据等。在传感器故障时，其具体位置可由红色闪烁的标识点得出，并且，用户能够在信息面板及时得到相应的故障告警。从图 5.10 和 5.11 中可以看出，所注入的传感器故障均能被构建的传感器故障诊断算法准确识别，并在原型系统的用户界面展示相应结果。在故障发生时，其他健康传感器的故障概率也会相应提高，但和故障传感器相比仍有较大差距，这符合第三章对传感器故障诊断算法的验证结果。

根据第四章所述的剩余使用寿命预测方法，针对该仿真实例搭建 DT-ResLSTM 剩余使用寿命预测算法并集成至该原型系统，在航程仿真模拟结束时，原型系统的机载子系统收集模拟数据并调用算法对该航程进行剩余使用寿命预测。预测结果如图 5.12 和 5.13 所示。

RUL Prediction & Component Situation Prediction

#	fan_eff	fan_flow	LPC_eff	LPC_flow	HPC_eff	HPC_flow	HPT_eff	HPT_flow	LPT_eff	LPT_flow	RUL
True	0	0	0	0	0	0	-0.002524	0	-0.001943	-0.003482	23
Prediction	-	-	-	-	-	-	-	-	-	-	21
Res	-	-	-	-	-	-	-	-	-	-	8.70%

图 5. 12 仿真实例剩余使用寿命预测结果（Web 端）

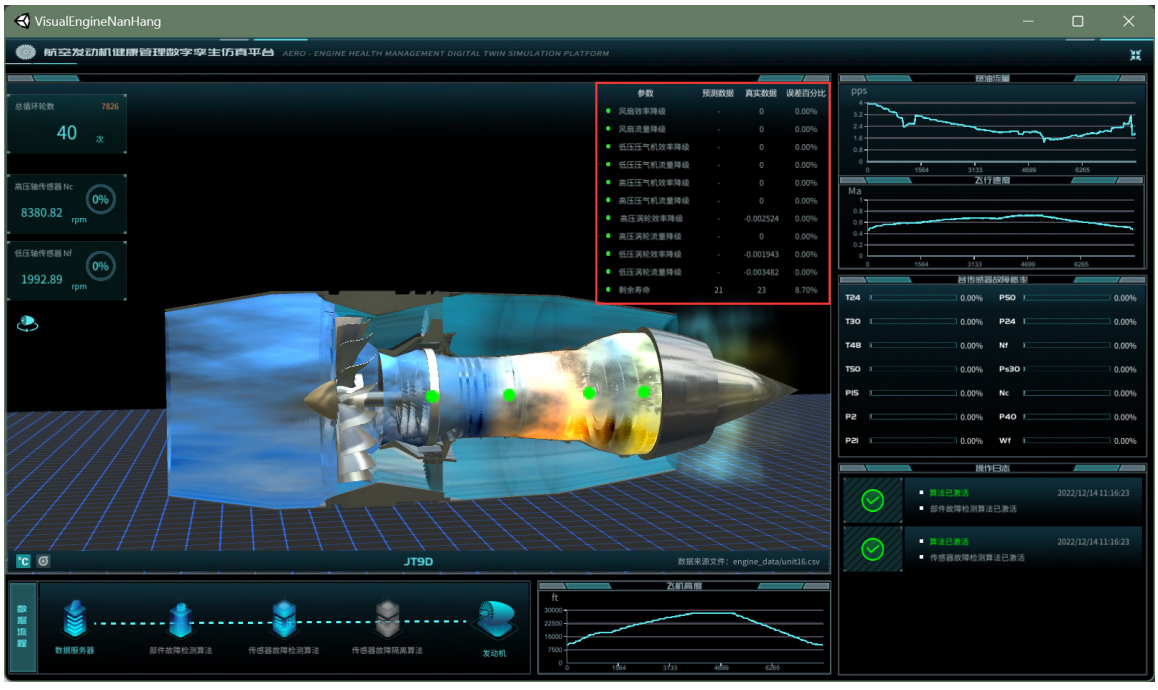


图 5. 13 仿真实例剩余使用寿命预测结果（Unity 三维可视化端）

图 5.12 和图 5.13 红色标识区域分别是该仿真实例在 Web 端和 Unity 三维可视化端的剩余使用寿命预测结果，用户从结果可得知该航程真实的部件退化情况、剩余使用寿命和算法预测的剩余使用寿命以及真实和预测之间的差距。从图中可知，该仿真实例的性能退化主要来源于高压涡轮和低压涡轮，真实和预测的剩余使用寿命分别为 23 次飞行航程，21 次飞行航程，预测和真实之间的相对误差为 8.70%，对于提供运维支持来说，该误差可以被接受。

5.8 本章小结

本章是综合考虑科研条件对第二章健康管理系统数字仿真平台方案设计的工程实践。本章首先介绍原型系统设计思路和整体功能架构，接着介绍了工程实现所需技术。然后着重对原型系统的机载子系统和地面子系统进行了工程设计，并且利用现有技术对原型系统进行了代码开发，最后通过子系统的部署情况和仿真实例对该原型系统进行了验证。