MA5233 Computational Mathematics

Lecture 0: Introduction

Simon Etter

Semester I, AY 2020/2021

Introduction

Mathematics is the language of nature.

Computational mathematics is our interpreter.

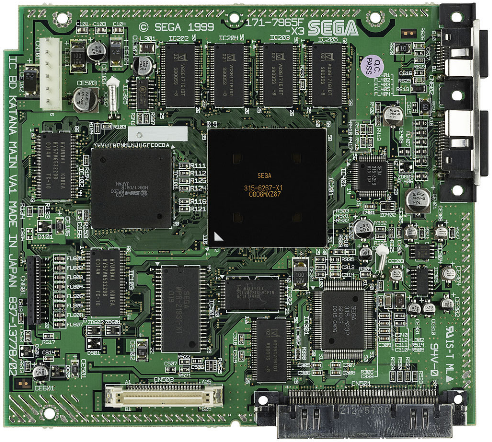
Introduction

Tentative module outline

- Fundamentals (weeks 1 - 5)
  - Machine arithmetic
  - Big-O notation
  - Consistency and stability
  - Nonlinear equations
  - Polynomial approximation theory
  - Quadrature
- Partial differential equations / sparse linear systems (weeks 6 - 10)
  - Finite difference discretisation
  - Sparse LU factorisation
  - Krylov subspace methods
  - Multigrid method
- Ordinary differential equations (weeks 11 - 12)
  - Explicit & implicit Runge-Kutta methods

Introduction

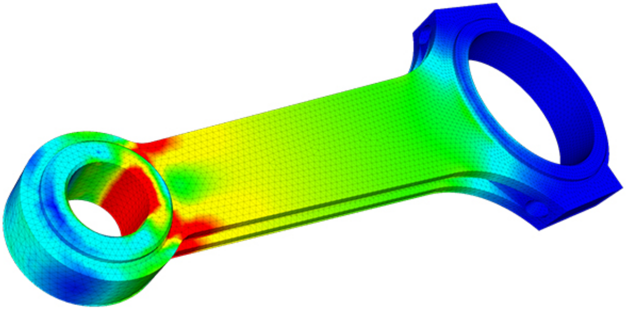
Fundamentals of computational mathematics



https://commons.wikimedia.org/wiki/File:Sega-Dreamcast-Motherboard-Top.jpg

Introduction

Partial differential equations / large, sparse linear systems



https://www.simscale.com/blog/2016/10/what-is-finite-element-method/

Introduction

Ordinary differential equations

https://pixabay.com/photos/rocket-launch-night-trajectory-693236/

Introduction

About this module

Challenges:

- I will cover many topics in a short amount of time.
  Expect to feel overwhelmed at times.
- A single three-hour class per week is not ideal.
  You will have to revise the lecture material after class to keep up.
- Diverse group of students.

Silver linings:

- The material presented in this module is highly relevant for many fulfilling and well-paying jobs.
- You are not alone!
  - Don't be shy to contact me (via LumiNUS $\rightarrow$ Chat Room) if I explain something poorly or you would like to know more.
  - Try to connect with your course mates, and try especially hard given the current circumstances. Learning is more fun when you are not alone, and having fun is the best motivation for working hard.

Introduction

Programming language

- We will be using the Julia programming language for demonstrations in class and the homework assignments.
- Julia offers many advantages compared to more well-known alternatives like Matlab or Python:
  - Modern and maths-friendly syntax.
  - Many state-of-the-art packages are easily available.
  - It is free to use even after you graduate.
- Julia is non-negotiable. You will have to learn Julia to pass this class. (Sorry about this, but any other solution would be worse.)
- Some help for getting started with Julia is provided under `https://github.com/ettersi/ComputationalMathematics/blob/master/julia.pdf`