TU Delft Light

An Easy to Use LaTeX Template

February 20th, 2020

TU Delft

Delft University of Technology

[TU0000] LaTeX 101

TU Delft Light

An Easy to Use LaTeX Template

Supervisors:
Dr. John Doe

Authors:
John Doe 0000001
Jane Doe 0000002

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

February 20, 2020

Table of Contents

List of Symbols

Abbreviations

ABCD	Ayy Bee See Dee

Roman Symbols

C_L	Lift Coefficient	$-$
V	Velocity	$\mathrm{kg\,m^{-1}}$
S	Wing Area	$\mathrm{m^{2}}$

Greek Symbols

ρ	Density of Air	$\mathrm{kg\,m^{-3}}$

List of Figures

List of Tables

1 Example LaTeX Elements

This template has been developed for the [AE3200] Design Synthesis Exercise by Şan Kılkış and Munyung Kim. The source code can be modified and redistributed but the license file must remain intact. Refer to the `LICENSE.md` included with the template for details.

1.1 Tables & Figures

An example Table 1.1 and an example Figure 1.1 can be found in this section. When you label tables or figures, make sure to use 'tab:name' or 'fig:name', this is not necessary for syntax but makes organization and look-up of labels easier. For inserting 2+ figures in a row, look at the formatting of Figure 1.2. Using the `cleveref` package negates the need for manually typing 'Table' or 'Figure'. The syntax is as follows, note that the 'tab' in 'tab:exampletable' is not necessary for `cref` and is purely for organizational reasons. However a ',' cannot be utilized as this is interpreted as a list.

```
\cref{tab:exampletable}
```

The Tables below use the package `tabularx` which adjusts column spacing automatically to fit the table within the margins of the page. The syntax is as follows where 'L' is for Left Aligned, 'C' for Centered, and 'R' is for Right Aligned:

```
\begin{tabularx}{\textwidth}{L C C C}
```

In order to keep up the same appearance for all tables use the commands `toprule`, `midrule`, `bottomrule`, and `hdashline` to create the horizontal lines. NO VERTICAL LINES ARE ALLOWED!

Table 1.1: Example Table

Component	**Mass [kg]**	**Location [m]**	**Location [% MAC]**
Wing	425.4	5.74	40.00
Main Landing Gear	243.1	5.82	45.00
Fuel System	80.74	5.91	50.00
Flight Control System	48.61	6.08	60.00
Hydraulics	4.660	6.08	60.00
Wing Group	**802.5**	**5.80**	**43.85**
Fuselage	265.2	5.74	40.00
Engine	409.4	1.64	-
Avionics	490.9	4.39	-
H. Tail	42.93	13.2	-
V. Tail	66.43	12.6	-
Nose Gear	54.58	2.50	-
Electrical	217.4	6.16	67.12
AC & Anti-Ice	215.7	6.16	67.12
Furnishings	241.5	6.16	67.12
Fuselage Group	**2004**	**5.01**	**-2.32**
OEW C.G.	**2806**	**5.24**	**10.88**

Table 1.2: Example Table II

m	$\Re\{\underline{\mathfrak{X}}(m)\}$	$-\Im\{\underline{\mathfrak{X}}(m)\}$	$\mathfrak{X}(m)$	$\frac{\mathfrak{X}(m)}{23}$	A_m	$\varphi(m)$ / °	φ_m / °
1	16.128	+8.872	16.128	1.402	1.373	-146.6	-137.6
2	3.442	-2.509	3.442	0.299	0.343	133.2	152.4
3	1.826	-0.363	1.826	0.159	0.119	168.5	-161.1
4	0.993	-0.429	0.993	0.086	0.08	25.6	90
5	1.29	+0.099	1.29	0.112	0.097	-175.6	-114.7
6	0.483	-0.183	0.483	0.042	0.063	22.3	122.5
7	0.766	-0.475	0.766	0.067	0.039	141.6	-122
8	0.624	+0.365	0.624	0.054	0.04	-35.7	90
9	0.641	-0.466	0.641	0.056	0.045	133.3	-106.3
10	0.45	+0.421	0.45	0.039	0.034	-69.4	110.9
11	0.598	-0.597	0.598	0.052	0.025	92.3	-109.3

Figure 1.1: TU Delft Logo Flame

(a) TU Delft Logo Flame

(b) TU Delft Logo Flame

Figure 1.2: Two Figures Side-by-Side

1.2 References & Citations

The biblatex package is used for references with the default 'numeric' style for in-text citations and references [1]. The references sorting style is set to 'none' meaning that the references are sorted by the order in which they appear in text. A sample file samplerefs.bib is included to help when dealing with different types of publications.

```
\cite{citationtag}
```

1.3 Equations & Nomenclature

When typesetting equations, you need to use a nomenclature code when you introduce a variable for the FIRST time, such that the variable is listed on the list of symbols. An example is given below by Equation 1.1. With the current implementation, duplicate nomenclature items are not automatically removed.

$$L = \frac{1}{2}\rho V^2 S \cdot C_L \tag{1.1}$$

The the list of symbols for the above equation were generated with the code below:

```
\nomenclature[A]{ABCD}{Ayy Bee See Dee}
\nomenclature[B]{$C_L$}{Lift Coefficient \nomunit{-}}
\nomenclature[B, 01]{$V$}{Velocity \nomunit{kg.m^{-1}}}
\nomenclature[B, 02]{$S$}{Wing Area \nomunit{m^{2}}}
\nomenclature[G]{$\rho$}{Density of Air \nomunit{kg.m^{-3}}}
```

1.4 Units and Numbers

To have uniform spacing and formatting of numbers and units the siunitx package can be used. The syntax for displaying a number with its corresponding unit as "5 kg" is as follows:

```
\SI{5}{\kilogram}
```

Formatting of a unit of measure as "kg" is as follows, pay close attention to the lower-case call to \si.

```
\si{\kilogram}
```

References

[1] Lots of Coffee and Caffiene. *LaTeX: A Lovely Typesetting Language*. No One Publishing House of Bravos, 2019.

A MATLAB Code

A.1 Optimization Run Case [RunCase.m]

```matlab
% Copyright 2018 San Kilkis, Evert Bunschoten
%
% Licensed under the Apache License, Version 2.0 (the "License");
% you may not use this file except in compliance with the License.
% You may obtain a copy of the License at
%
%    http://www.apache.org/licenses/LICENSE-2.0
%
% Unless required by applicable law or agreed to in writing, software
% distributed under the License is distributed on an "AS IS" BASIS,
% WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
% See the License for the specific language governing permissions and
% limitations under the License.

classdef RunCase < handle
    %RUNCASE Summary of this class goes here
    %   Detailed explanation goes here

    properties
        aircraft;           % Aircraft instance with all parameters and vars
        x;                  % DesignVector object for fmincon & ease of use
        x_final;            % Optimized Design Vector
        converged = false;  % True if fmincon stopped w/o errors
        options;            % fmincon options struct
    end

    properties (SetAccess = private, GetAccess = public)
        cache = struct()    % Cache of Results & Constraints
        run_parallel        % Bool, True for machines with >= 4 cores
        iter_counter = 0    % Counts the number of function calls
        start_time          % datetime at the start of optimization
        end_time            % datetime at the end of optimization
        sim_time;           % Total Sim. Time at end of Optimization [s]
    end

    methods

        function obj = RunCase(aircraft_name, options)
            % Displaying welcome message
            type data\log\header.txt; fprintf('\n')

            % Constructing the specified aircraft
            obj.aircraft = aircraft.Aircraft(aircraft_name);
            obj.init_design_vector(); % Creating the design vector object

            % Augmenting options w/ OutputFnc
            options.OutputFcn = @obj.cache_optimValues;
            obj.options = options;
            obj.cache.results = []; % Results caching
            obj.cache.fmincon = []; % Solver caching
            obj.cache.const = [];   % Constraint caching
            obj.cache.time = [];    % Log of analysis time
        end

        function init_design_vector(obj)
            dv = @optimize.DesignVector;
            ac = obj.aircraft;
            obj.x = dv({'lambda_1', ac.lambda_1, 0, 1.25;...
                        'lambda_2', ac.lambda_2, 0.94, 1.25;...
                        'b', ac.b, 0.71, 1.06;...
                        'c_r', ac.c_r, 0.68, 1.15;...
                        'tau', ac.tau, 0.16, 2.5;...
                        'A_root', ac.A_root', 0.5, 1.2;...
                        'A_tip', ac.A_tip', 0.5, 1.2;...
                        'beta_root', ac.beta_root, 0, 1.7;...
                        'beta_kink', ac.beta_kink, -0.8, 3.2;...
                        'beta_tip', ac.beta_tip, -3.6, 3.6;...
                        % Get these values from first initial run
                        'A_L', ac.A_L, -1.5, 1.5;...
                        'A_M', ac.A_M, -1.5, 1.5;...
                        'W_w', ac.W_w, 0.6, 1.0;...
                        'W_f', ac.W_f, 0.6, 1.0;...
                        'C_d_w', ac.C_d_w, 0.8, 1.0});
        end

        function optimize(obj)
            obj.start_time = datetime(); tic;
            n_cores = feature('numcores');

            % Launching either in Parallel or Serial Execution
            try
                if n_cores >= 4
                    parpool(4)
                    obj.run_parallel = true;
                end
            catch
                obj.run_parallel = false;
                warning(['Parallel Processing Disabled ' ...
                         'or not Installed on Machine. Optimization '...
                         'will execute as a serial process!'])
            end

            [opt, ~] = fmincon(@obj.objective,...
                            obj.x.vector, [], [], [], [],...
                            obj.x.lb, obj.x.ub, @obj.constraints,...
                            obj.options);

            obj.sim_time = toc;
            obj.x_final = opt;
            obj.end_time = datetime();
            obj.converged = true;
            obj.shutdown();
        end

        function [c, ceq] = constraints(obj, x)
            disp('Constraints')
            res = obj.fetch_results(x);
            Cons = optimize.Constraints(obj.aircraft, res, obj.x);
            c = Cons.C_ineq; ceq = Cons.C_eq;

            % Caching of constraints
            if isempty(obj.cache.const)
                obj.cache.const.c = c;
                obj.cache.const.ceq = ceq;
            else
                obj.cache.const.c(end+1, :) = c;
                obj.cache.const.ceq(end+1, :) = ceq;
            end
        end

```

```matlab
        function fval = objective(obj, x)
            disp('Access from objective')
            res = obj.fetch_results(x);
            fval = res.W_f/obj.x.W_f_0;
        end

        function res = fetch_results(obj, x)
            if ~obj.x.isnew(x) && ~isempty(obj.cache.results)
                res = obj.cache.results(end);
            else
                disp('I asked for new runs')
                obj.x.vector = x; % Updates design vector w/ fmincon value
                obj.aircraft.modify(obj.x);
                obj.iter_counter = obj.iter_counter + 1;
                % Running Analysis Blocks
                if obj.run_parallel
                    tic;
                    spmd
                        if labindex == 1
                            temp = obj.run_aerodynamics();
                        elseif labindex == 2
                            temp = obj.run_structures();
                        elseif labindex == 3
                            temp = obj.run_loads();
                        elseif labindex == 4
                            temp = obj.run_performance();
                        end
                    end
                    t = toc;
                    fprintf('Parallel Process took: %.5f [s]\n', t)
                    res.C_dw = temp{1};
                    res.Struc = temp{2};
                    res.Loading = temp{3};
                    res.W_f = temp{4};
                else
                    tic;
                    res.C_dw = obj.run_aerodynamics();
                    res.Loading = obj.run_loads();
                    res.Struc = obj.run_structures();
                    res.W_f = obj.run_performance();
                    t = toc;
                end

                if isempty(obj.cache.results)
                    obj.cache.results = res;
                    obj.cache.time = t;
                else
                    obj.cache.results(end+1) = res;
                    obj.cache.time(end+1) = t;
                end
            end
        end

        function A = run_aerodynamics(obj)
            try
                Aero = aerodynamics.Aerodynamics(obj.aircraft);
                A = Aero.C_d_w;
            catch
                A.C_D_w = NaN;
            end
        end

        function L = run_loads(obj)
```

```
            try
                Loads = loads.Loads(obj.aircraft);
                L.M_distr = Loads.M_distr;
                L.L_distr = Loads.L_distr;
                L.Y_coord = Loads.Y_coord;
            catch
                L.M_distr = ones(length(obj.x.A_M),1) * NaN;
                L.L_distr = ones(length(obj.x.A_M),1) * NaN;
                L.Y_coord = NaN;
            end
        end

        function S = run_structures(obj)
            try
               Structures = structures.Structures(obj.aircraft);
               S.W_w = Structures.W_w;
               S.V_t = Structures.V_t;
            catch
                S.W_w = NaN;
                S.V_t = NaN;
            end
        end

        function P = run_performance(obj)
            try
                perf = performance.Performance(obj.aircraft);
                P = perf.W_fuel;
            catch
                P.W_fuel = NaN;
            end
        end

        function stop = cache_optimValues(obj, x, optimValues, state)
            stop = false;
            o = optimValues;
            switch state
                case 'init'
                    % hold on
                    obj.cache.fmincon = o;
                    obj.cache.fmincon.x = x;
                case 'iter'
                    % Concatenate current point and objective function
                    % value with history. x must be a row vector
                    history = obj.cache.fmincon;
                    temp.fval = [history.fval, o.fval];
                    temp.x = [history.x, x];

                    % Gradient caching of fmincon
                    temp.gradient = [history.gradient,...
                                     o.gradient];

                    % Optimality caching of fmincon
                    temp.firstorderopt = [history.firstorderopt,...
                                          o.firstorderopt];

                    temp.iteration = [history.iteration, o.iteration];
                    temp.funccount = [history.funccount, o.funccount];
                    obj.cache.fmincon = temp;

                case 'done'
                    % hold off
                otherwise
            end
        end

        function shutdown(obj)
            if obj.run_parallel
                % Shutting Down Parallel Pool
                poolobj = gcp('nocreate');
                delete(poolobj);
            end
            obj.end_time = datetime();
            if isempty(obj.sim_time)
                obj.sim_time = obj.end_time - obj.start_time;
            end
        end
    end

    methods (Static)
         function obj = load_run(run_file)
            filename = [pwd '\data\runs\' run_file '.mat'];
            try
                loaded_obj = load(filename, 'run_case');
                obj = loaded_obj.run_case;
            catch
                error('Supplied file has no property: run_case')
            end
        end
    end
end
```