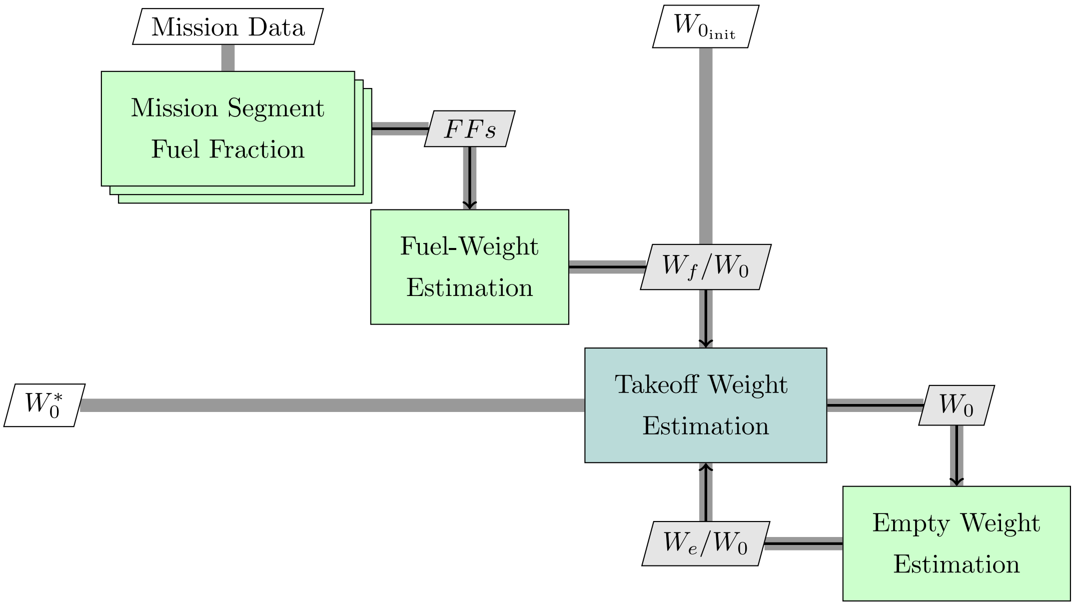
Initial Weight Estimation



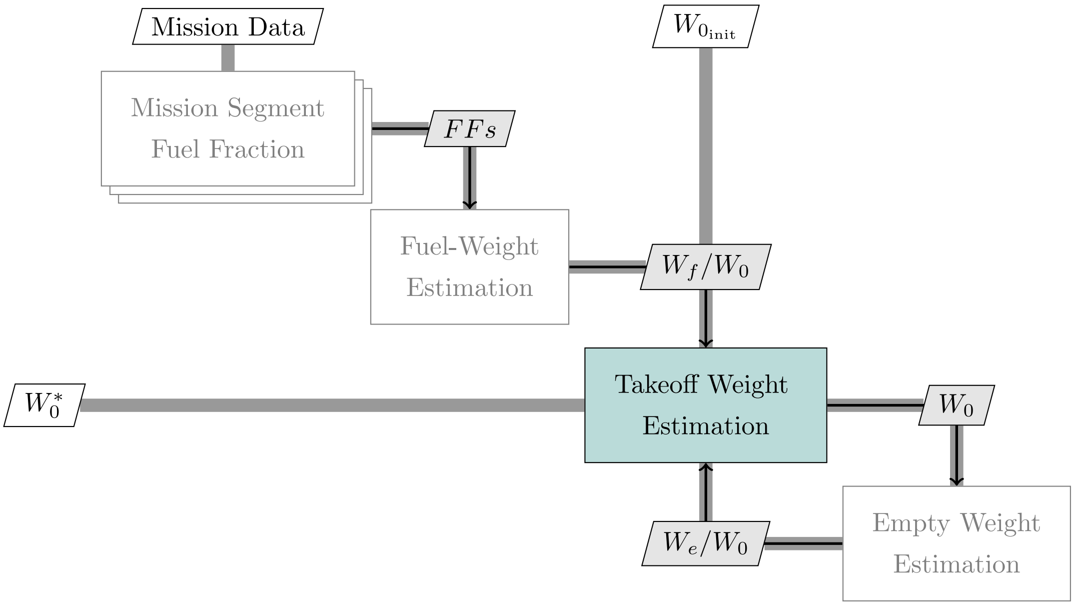
Aircraft specifications:

- Crew: 5 passengers, 70 kgs
- Payload: Radar equipment, 450 kgs (still in revision)
- Cruise condition: $M = 0.2354$ at $10,000$ ft
- Maximum lift-to-drag ratio: $(L/D)_{\max} = 10 - 12$

```
12.0
```

```
begin
    g = 9.81
    WPL = 450g
    Wcrew = 350g
    M = 0.2354
    LD_max = 12.0
end
```

Maximum Takeoff Weight



The 'governing equation' of this problem is:

$$W_0 = \frac{W_{\text{payload}} + W_{\text{crew}}}{1 - \dfrac{W_f}{W_0} - \dfrac{W_e}{W_0}}$$

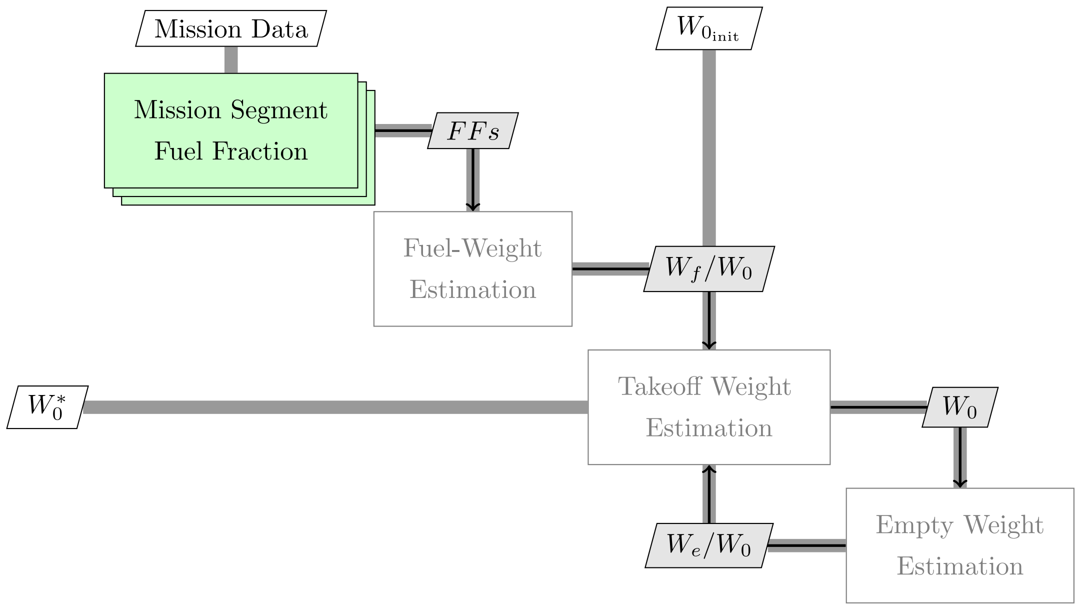
```
maximum_takeoff_weight (generic function with 1 method)
```

```julia
maximum_takeoff_weight(WPL, Wcrew, WfWTO, WeWTO) = (WPL + Wcrew)/(1 - WfWTO - WeWTO)
```

```
takeoff_weight (generic function with 1 method)
```

```julia
function takeoff_weight(WPL,Wcrew,WfWTO,WeWTO)
    return (WPL + Wcrew)/(1 - WfWTO - WeWTO)
end
```

Mission Fuel Fractions



Takeoff

```
takeoffWF = 0.998;
```

Climb

Enter a variable name you would like to use for the climb weight fraction, and assign it an associated value.

```
climbWF = 0.992;
```

Cruise

$$WF_{\text{cruise}} = \exp\left(-\frac{R \times SFC}{V \times (L/D)_{\text{cruise}}}\right)$$

cruiseWF_func (generic function with 1 method)

```
cruiseWF_func(range,LD,V,SFC) = exp(-range*SFC/(V*LD))
```

The SFC of this aircraft configuration at cruise is: 0.5 - 0.7 1/hr with added efficiency of 0.8

```
begin
    SFC = 0.7
    LD_cruise = 9*LD_max
    R1 = 700*1852
    c = 327.7916 #@ 10,000 ft
    V = M*c
    cruise_SFC = SFC/3600
end;

```

cruise1WF = 0.9702042264043564

```
cruise1WF = cruiseWF_func(R1,LD_cruise,V, cruise_SFC)
```

R2 = 1881000

```
R2 = 1000*1881
```

cruise2WF = 0.9570601257750827

```
cruise2WF = cruiseWF_func(R2,LD_cruise,V, cruise_SFC)
```

Loiter

$$WF_{\text{loiter}} = \exp\left(-\frac{E \times SFC}{(L/D)_{\max}}\right)$$

The SFC of this aircraft configuration at loiter is: $0.5 - 0.7$ 1/hr

loiterWF_func (generic function with 1 method)

```
loiterWF_func(E,SFC, LD) = exp(-E*SFC / (LD))
```

```
begin
    E1 = 4 * 3600
    loiter_SFC = 0.5/3600
end;
```

loiter1WF = 0.846481724890614

```
loiter1WF = loiterWF_func(E1, loiter_SFC, LD_max)
```

E2 = 5400.0

```
E2 = 1.5 * 3600
```

loiter2WF = 0.9394130628134758

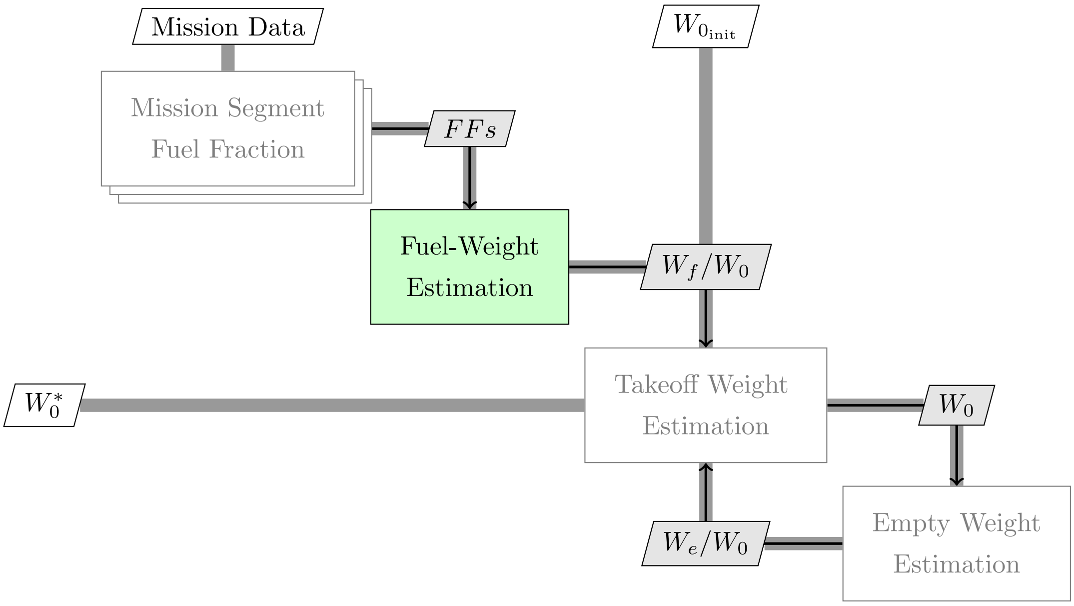
```
loiter2WF = loiterWF_func(E2, loiter_SFC, LD_max)
```

Landing

$$WF_{\text{landing}} = 0.993$$

```
landingWF = 0.993;
```

Fuel Weight Fractions



$$\frac{W_f}{W_0} = a\left(1 - \prod_{i=1}^{N} \frac{W_{f_i}}{W_{f_{i-1}}}\right)$$

```
fuelWF_func (generic function with 1 method)
```

```
fuelWF_func(a,ratios) = a * (1-prod(ratios))
```

Assign your computed weight fractions to an array.

```
FFs = Float64[0.998, 0.992, 0.970204, 0.846482, 0.95706, 0.939413, 0.993]
```

```
FFs = [takeoffWF, climbWF, cruise1WF, loiter1WF, cruise2WF, loiter2WF, landingWF]
```

Fuel fraction has a reserve fuel requirement of 6%

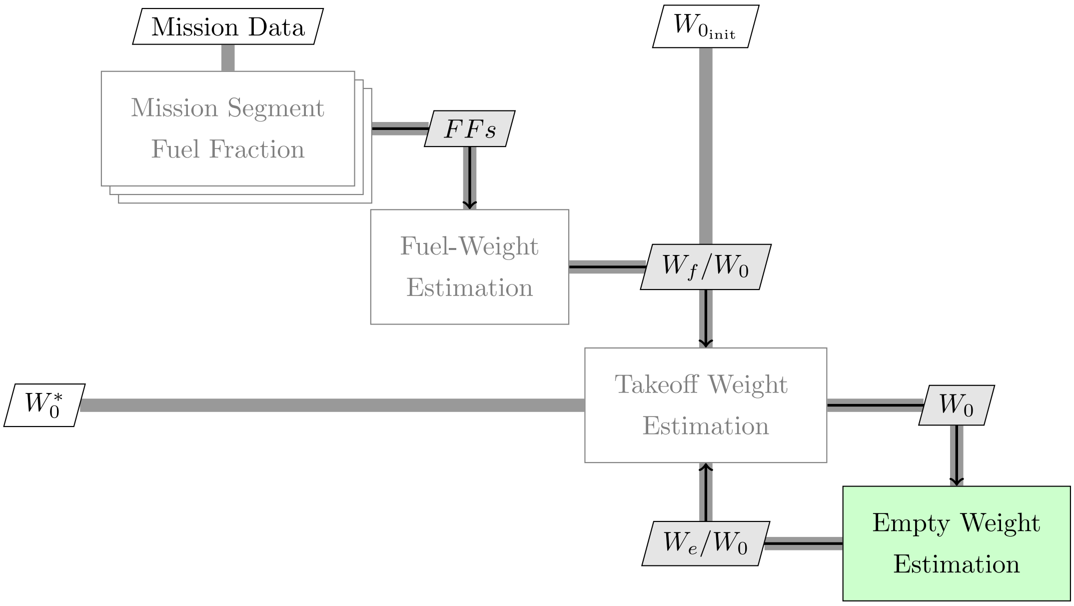
```
a = 1.06
```

```
a = 1.06
```

```
WfWTO = 0.2905615354215037
```

```
WfWTO = fuelWF_func(a,FFs)
```

Empty Weight Fraction



Raymer's regression formula:

$$\frac{W_e}{W_0} \equiv W_{EF}(W_0) = AW_0^B$$

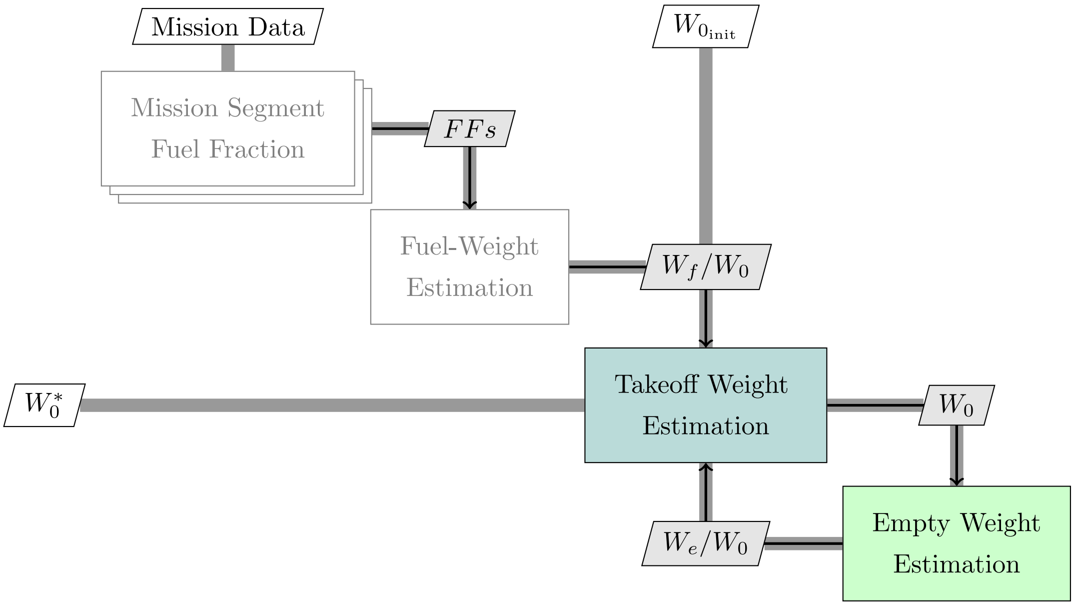
```
empty_weight_raymer (generic function with 1 method)
```

```
empty_weight_raymer(WTO, A, B) = A * WTO^B
```

Raymer regression coefficients:

```
begin
    A = -0.144
    B = 1.1162
end;
```

Iterative Estimation



Fixed-Point Iteration

Given:

$$\begin{aligned} \text{Fuel Weight Fraction:} \quad & \frac{W_f}{W_0} = a\left(1 - \prod_{i=1}^{N} \frac{W_{f_i}}{W_{f_{i-1}}}\right) \\ \text{Empty Weight Fraction:} \quad & \frac{W_e}{W_0} = AW_0^B \end{aligned}$$

We need to solve the following equation for W_0:

$$W_0 = \frac{W_{\text{payload}} + W_{\text{crew}}}{1 - \dfrac{W_f}{W_0} - \dfrac{W_e}{W_0}}$$

We can express as the equality of two expressions. Here, we will denote the iteration number of a variable by an additional subscript $(-)_n$:

$$\begin{aligned} \text{Takeoff Weight Expression:} \quad & (W_0)_n = \frac{W_{\text{payload}} + W_{\text{crew}}}{1 - \dfrac{W_f}{W_0} - \left(\dfrac{W_e}{W_0}\right)_{n-1}} \\ \text{Equality:} \quad & (W_0)_n = (W_0)_{n-1} \end{aligned}$$

Let the following indicate the relative error for the nth iteration:

$$\varepsilon_n = \left| \frac{(W_0)_n - (W_0)_{n-1}}{(W_0)_{n-1}} \right|$$

We would like our analysis to converge below some tolerance ε_{tol}, i.e. the error should be $\varepsilon_n < \varepsilon_{\text{tol}}$.

```
compute_mtow (generic function with 1 method)
```

```
function compute_mtow(W_0, W_PL, W_crew, WfWTO, A, B; num_iters = 20, tol = 1e-12)
```

```
    # Initial value (guess)
    WTO = W_0

    # Array of takeoff weight values
    WTOs = [WTO]

    # Array of errors over iterations of size num_iters, initially infinite
    errors = [ Inf; zeros(num_iters) ]

    # Iterative loop
    for i in 2:num_iters

        # Empty weight fraction
        WeWTO = empty_weight_raymer(WTO, A, B)

        # Maximum takeoff weight
        new_WTO = maximum_takeoff_weight(W_PL, W_crew, WfWTO, WeWTO)

        # Evaluate relative error
        error = abs((new_WTO - WTO)/WTO)

        # Append WTO to end of WTOs array
        push!(WTOs, WTO)

        # Assign error to errors array at current index
        errors[i] = error

        # Conditional
        if error < tol
            break
        else
            # Assign new takeoff weight to WTO
            WTO = new_WTO
        end
    end

    # Return arrays of takeoff weights and errors
    WTOs, errors[1:length(WTOs)]
end
```

```
Float64[0.0, 0.0, 0.0, 0.0]
```

```
zeros(4)
```

Exercise

Rewrite this function using a `while` loop instead of a `for` loop.

```
compute_mtow_while (generic function with 1 method)
```

```
function compute_mtow_while(W_0, W_PL, W_crew, WfWTO, A, B; num_iters = 20, tol =
1e-12)
    # Initial value (guess)
    WTO_while = W_0

    # Array of takeoff weight values
    WTOs_while = [WTO_while]

    # Array of errors over iterations of size num_iters, initially infinite
    errors_while = [ Inf; zeros(num_iters) ]

    j = 2

    # Empty weight fraction
```

```
        WeWTO = empty_weight_raymer(WTO_while, A, B)

        # Maximum takeoff weight
        new_WTO_while = maximum_takeoff_weight(W_PL, W_crew, WfWTO, WeWTO)

        # Evaluate relative error
        error_while = abs((new_WTO_while - WTO_while)/WTO_while)

    while error_while > tol

        push!(WTOs_while,WTO_while)

        j = j + 1

        errors_while[j] = error_while
        if j > num_iters
            break
        else
            WTO_while = new_WTO_while
        end
    end

    WTOs_while, errors_while[1:length(WTOs_while)]
end
```

Set an initial value for the takeoff weight estimation procedure:

```
W0 = 7848.0
```

```
W0 = WPL + Wcrew
```

Run your `compute_mtow()` function here with the relevant inputs:

```
(Float64[7848.0,  7848.0,  2.4489,  7129.63,  2.72581,  6821.6,  2.86352,  6676.85,  2.932
```

```
WTOs, errors = compute_mtow(W0, WPL, Wcrew, WfWTO, A, B;
    num_iters = 20, tol = 1e-12)
```

```
(Float64[7848.0,  7848.0,  2.4489,  2.4489,  2.4489,  2.4489,  2.4489,  2.4489,  2.4489,
```

```
WTOs_while, errors_while = compute_mtow_while(W0, WPL, Wcrew, WfWTO, A, B;
    num_iters = 20, tol = 1e-12)
```

Check the final value of the maximum takeoff weight.

```
md"Check the final value of the maximum takeoff weight."
```

```
empty (generic function with 1 method)
```

```
function empty(W_0, W_PL, W_crew, WfWTO, A, B; num_iters = 20, tol = 1e-12)
    # Initial value (guess)
    WTO_while = W_0

    # Array of takeoff weight values
    WTOs_while = [WTO_while]

    # Array of errors over iterations of size num_iters, initially infinite
    errors_while = [ Inf; zeros(num_iters) ]

```

```
    j = 2

    # Empty weight fraction
        WeWTO = empty_weight_raymer(WTO_while, A, B)

        # Maximum takeoff weight
        new_WTO_while = maximum_takeoff_weight(W_PL, W_crew, WfWTO, WeWTO)

        # Evaluate relative error
        error_while = abs((new_WTO_while - WTO_while)/WTO_while)

    while error_while > tol

        push!(WTOs_while,WTO_while)

        j = j + 1

        errors_while[j] = error_while
        if j > num_iters
            break
        else
            WTO_while = new_WTO_while
        end
    end

    WeWTO
end
```

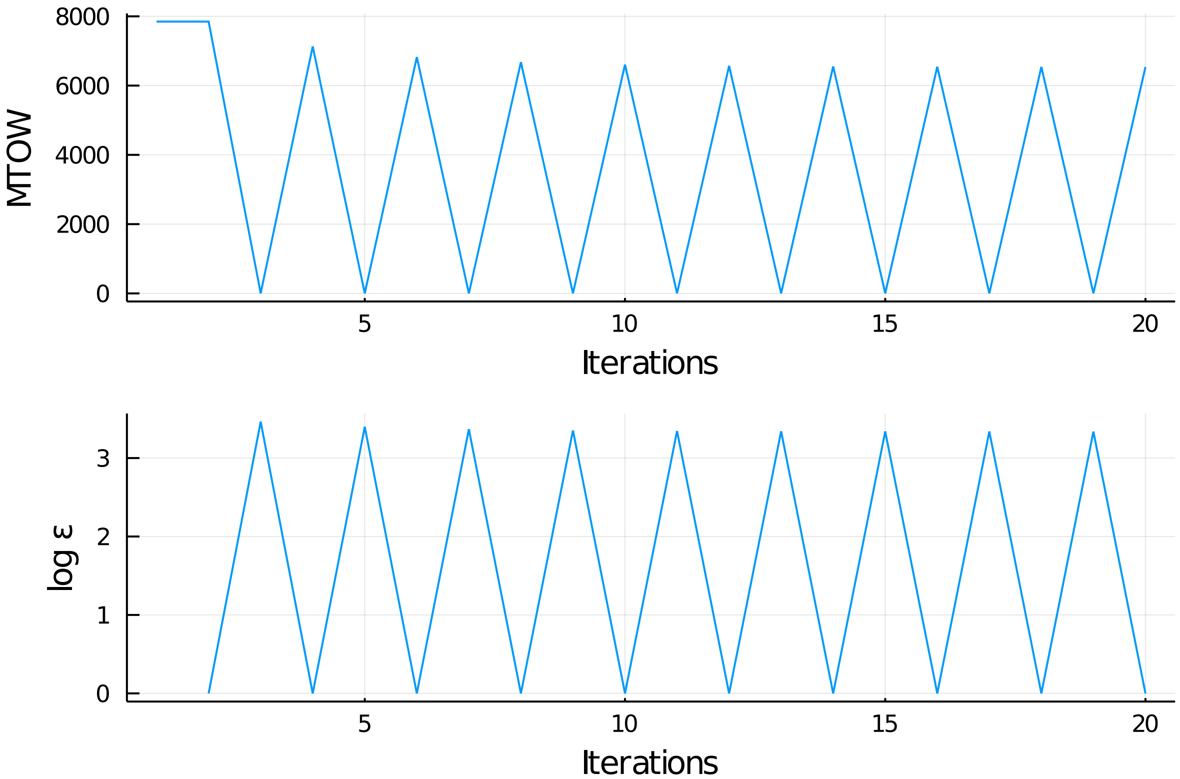
```
WeWTO = -3203.999809668005
```

```
WeWTO = empty(W0, WPL, Wcrew, WfWTO, A, B;
    num_iters = 20, tol = 1e-12)
```

```
Ans = 2.4488961064324664
```

```
Ans = maximum_takeoff_weight(WPL, Wcrew, WfWTO, WeWTO)
```

```
Enter cell code...
```



```julia
begin

    plot1 = plot(WTOs,
                 label = :none, ylabel = "MTOW", xlabel = "Iterations")
    plot2 = plot(log10.(errors),
                 label = :none, ylabel = "log ε", xlabel = "Iterations")

    plot(plot1, plot2, layout = (2,1))
end
```

Investigation

The following exercise is not graded, and is for you to investigate the effects of the regression formula used for the empty weight fraction.

Exercise

Write a function that computes the empty weight using Roskam's regression formula:

$$W_e = 10^{(\log_{10} W_0 - C)/D}$$

Warning!

The coefficients C and D are not the same as A and B as shown previously in Raymer's formula!

Hint

[illegible]

Reuse this function in your takeoff weight estimation function `compute_mtow()` by computing the empty weight fraction and see if you get different results!

```
ArgumentError: Package PlutoUI not found in current path:
- Run `import Pkg; Pkg.add("PlutoUI")` to install the PlutoUI package.

  1. require(::Module, ::Symbol) @ loading.jl:893
  2. top-level scope @ Local: 2
```