In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
from IPython.display import display, Markdown

skip_plots = False
```

In [ ]:
```
def ias2tas(ias, alt):
  return ias + (alt/10.0 * (ias * 0.02))
```

In [ ]:
```
matrix = {}

def parseFile(file):
  weights = [140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290]
  matrix = {}
  file = open(file, 'r')
  lines = file.readlines()

  count = 0
  for line in lines:
    count += 1
    if count == 1:
      continue
    fields = line.split(";")
    alt = int(fields[0]) / 100
    ff = [int(item.strip()) for item in fields[1].split(",")]
    ias = [int(item.strip()) for item in fields[2].split(",")]
    mach = [float(item.strip()) for item in fields[3].split(",")]

    inner = {}
    for i in range(len(ff)):
      _ias = ias[i] if len(ias) > 1 else ias[0]
      _mach = mach[i] if len(mach) > 1 else mach[0]
      inner[weights[i]] = (ff[i], _ias, _mach)
    matrix[alt] = inner
  return matrix

matrix["LRC"] = parseFile("LRC.data")
matrix[82] = parseFile(".82.data")
matrix[83] = parseFile(".83.data")
matrix[84] = parseFile(".84.data")
```

In [ ]:
```
points_for_fit = 1000
```

In [ ]:
```
def func_exp(x, a, b, c):
  return a * np.exp(b * x) + c * x

def func_quad(x, a, b, c):
  return a * x**2 + b * x + c

def func_root(x, a, b, c):
  return a * np.sqrt(x) + b * x + c
```

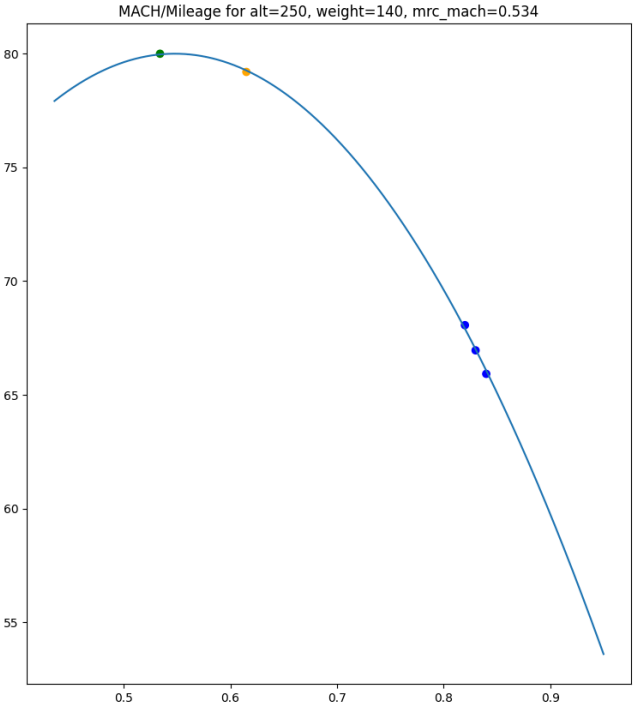
In [ ]:
```
def get_lrc_mach(weight, alt):
  return matrix["LRC"][alt][weight][2]

def generate_base_lists(weight, alt, mrc_mach):
  raw = []
  col = []
  try:
    for spd in range(82, 85):
      pt = matrix[spd][alt][weight]
```

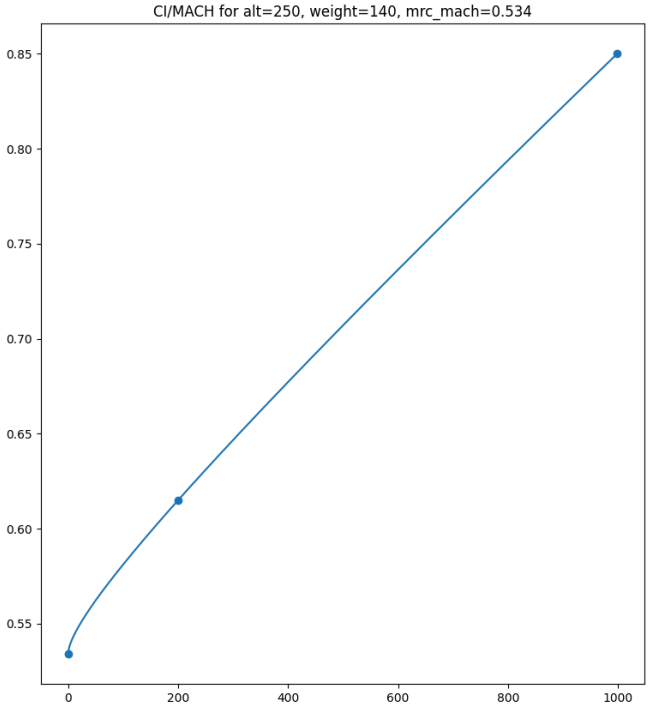
```
      raw.append((pt[0], ias2tas(pt[1], alt), pt[2]))
  except KeyError:
    pass

  lrc = matrix["LRC"][alt][weight]
  raw.append((lrc[0], ias2tas(lrc[1], alt), lrc[2], "LRC"))
  raw.sort(key=lambda i: i[2])

  spd_base = np.array([x[1] for x in raw])
  ff_base = np.array([x[0] * 3 for x in raw])
  mileage_base = spd_base / ff_base * 1000
  index = next((i for i, item in enumerate(raw) if len(item) == 4 and item[3] == 'LRC'), -10)
  lrc_mileage = mileage_base[index]
  mrc_mileage = lrc_mileage * 100.0/99.0

  raw.append((-1, -1, mrc_mach, "MRC"))
  raw.sort(key=lambda i: i[2])
  for i in range(len(raw)):
     if len(raw[i]) == 4 and raw[i][3] == "MRC":
        mileage_base = np.insert(mileage_base, i, mrc_mileage)
        break

  for x in raw:
    if len(x) < 4:
      col.append("blue")
    elif x[3] == "MRC":
      col.append("green")
    elif x[3] == "LRC":
      col.append("orange")
    else:
      col.append("red")

  mach_base = np.array([x[2] for x in raw])
  return mach_base, mileage_base, col, mrc_mileage, mrc_mach

def find_best_mrc_mach(weight, alt, show_intermediate_graphs = False, debug_output = False):
  lrc_mach = get_lrc_mach(weight, alt)
  rng = np.linspace(lrc_mach - 0.1, lrc_mach, 100, False)
  best_mrc_mach = 0
  best_func = 0
  last_mrc_mileage_error = 10000

  for mrc_mach in rng:
    mach, mileage, colour, mrc_mileage, mrc_mach = generate_base_lists(weight, alt, mrc_mach)
    x = np.linspace(mrc_mach - 0.1, 0.95, points_for_fit)

    popt, pcov = sp.optimize.curve_fit(func_quad, mach, mileage, maxfev=100000)
    mileage_mach_func = func_quad(x, *popt)

    if debug_output:
      print(f"{alt=}, {weight=}, {mrc_mach=}", mrc_mileage, max(mileage_mach_func))

    if abs(mrc_mileage - max(mileage_mach_func)) < last_mrc_mileage_error:
      last_mrc_mileage_error = abs(mrc_mileage - max(mileage_mach_func))
      best_mrc_mach = round(mrc_mach, 3)
      best_func = mileage_mach_func
    else:
      return best_mrc_mach, lrc_mach, x, best_func

    if show_intermediate_graphs and not skip_plots:
      plt.figure(figsize=(20,10))
      plt.title(f"MACH/Mileage for {weight=} and {mrc_mach=}")
      plt.plot(x, mileage_mach_func)
      plt.scatter(mach, mileage, c=colour)
      plt.show()
```

In [ ]:
```
curves = []
for alt in range(250, 450, 20):
  _curves = []
  for weight in range(140, 300, 10):
    try:
      best_mrc_mach, lrc_mach, x, mileage_mach_func = find_best_mrc_mach(weight, alt)
      display(Markdown(f"**Found: {best_mrc_mach} MRC and {lrc_mach} LRC**"))

      mach, mileage, colour, mrc_mileage, mrc_mach = generate_base_lists(weight, alt, best_mr

      if not skip_plots:
        plt.figure(figsize=(20,10))

        plt.subplot(1, 2, 1)
        plt.title(f"MACH/Mileage for {alt=}, {weight=}, {mrc_mach=}")
        plt.plot(x, mileage_mach_func)
        plt.scatter(mach, mileage, c=colour)

      ci_mach_base = [best_mrc_mach, lrc_mach, 0.85]
      ci_base = [0, 200, 999]
      x = np.linspace(0, 999, points_for_fit)

      popt, pcov = sp.optimize.curve_fit(func_root, ci_base, ci_mach_base, maxfev=10000)
      ci_mach_func = func_root(x, *popt)

      if not skip_plots:
        plt.subplot(1, 2, 2)
        plt.title(f"CI/MACH for {alt=}, {weight=}, {mrc_mach=}")
        plt.scatter(ci_base, ci_mach_base)
        plt.plot(x, ci_mach_func)
        plt.show()

      display(Markdown(f"**Sanity check: {ci_mach_func[0]} MRC and {round(ci_mach_func[int(po

      _curves.append(ci_mach_func)
    except KeyError:
      pass
  curves.append(_curves)
```

Found: 0.534 MRC and 0.615 LRC

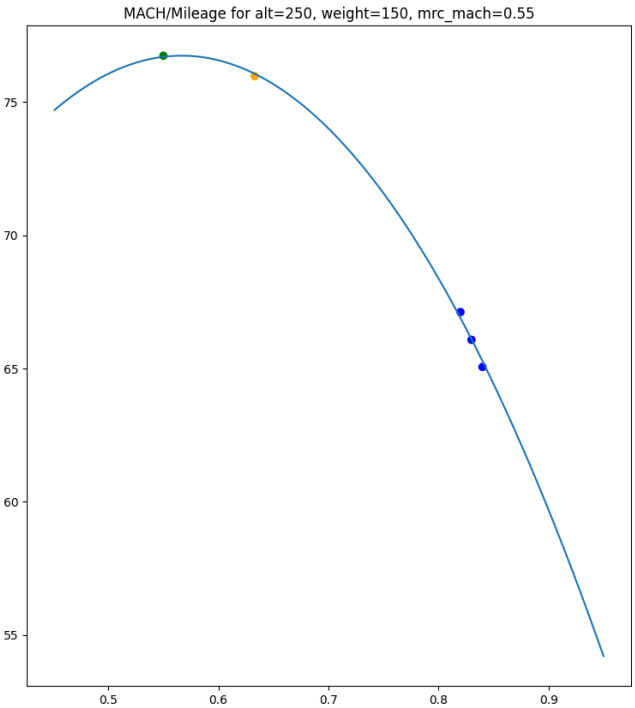
```
C:\Users\llego\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\Local
Cache\local-packages\Python311\site-packages\scipy\optimize\_minpack_py.py:1010: OptimizeWarni
ng: Covariance of the parameters could not be estimated
  warnings.warn('Covariance of the parameters could not be estimated',
```

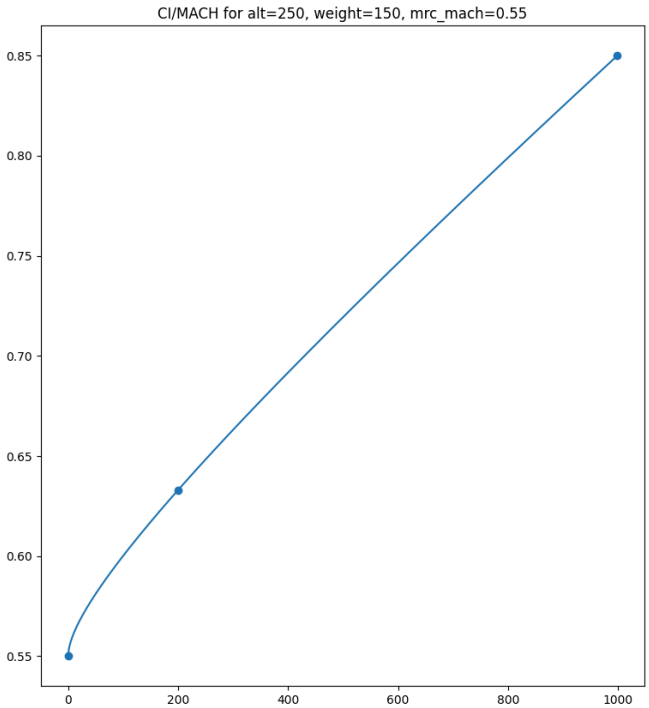




Sanity check: 0.534 MRC and 0.615 LRC

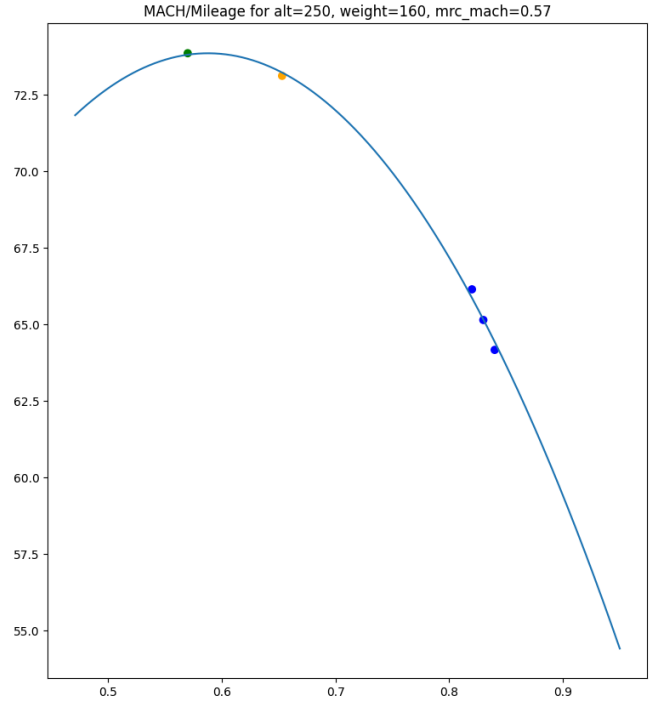
Found: 0.55 MRC and 0.633 LRC

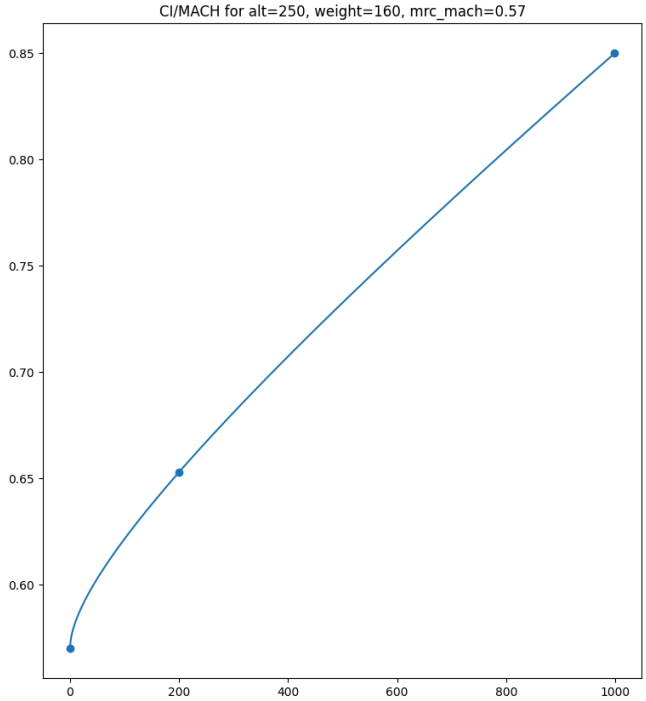




Sanity check: 0.55 MRC and 0.633 LRC

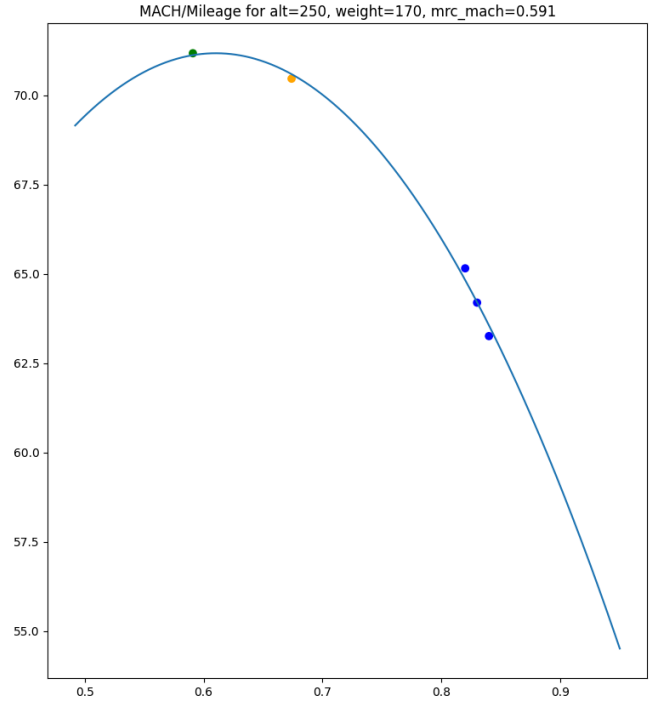
Found: 0.57 MRC and 0.653 LRC

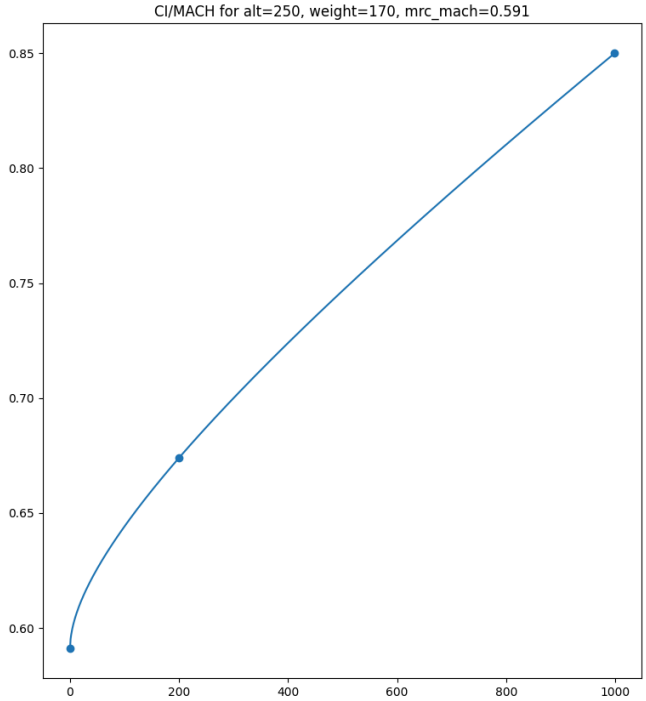




Sanity check: 0.57 MRC and 0.653 LRC

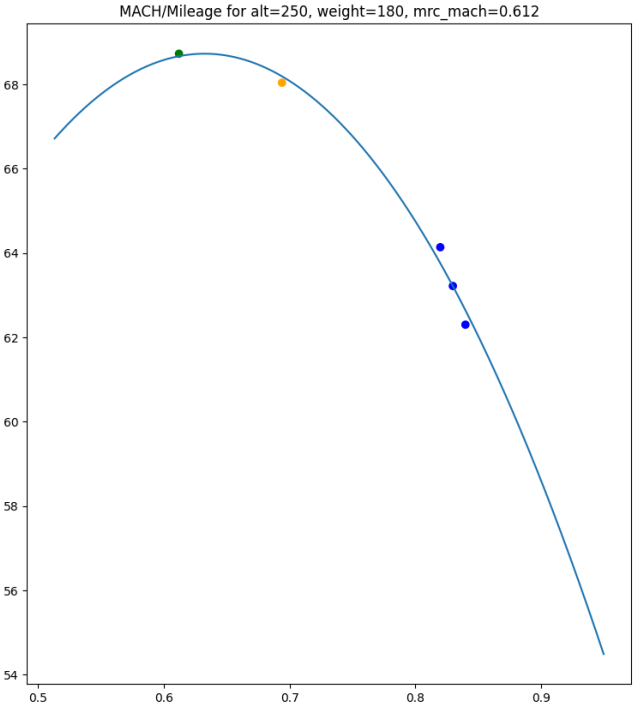
Found: 0.591 MRC and 0.674 LRC

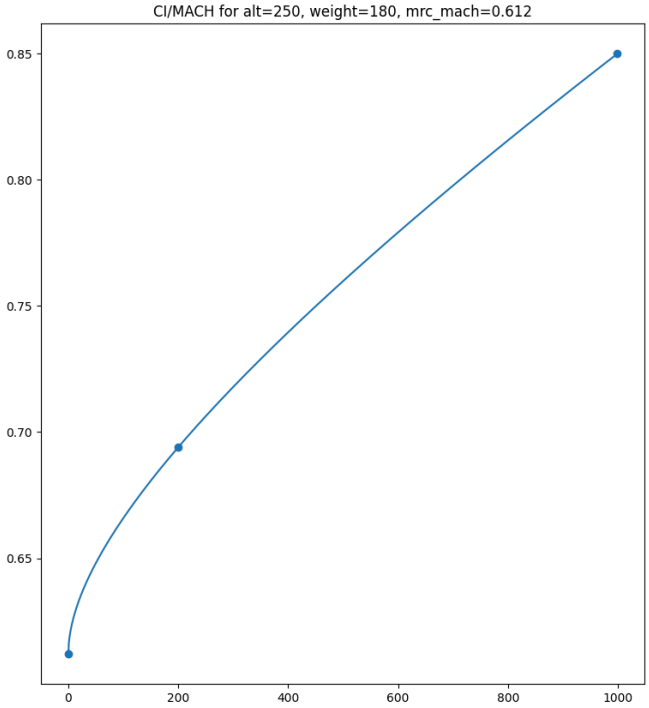




Sanity check: 0.591 MRC and 0.674 LRC

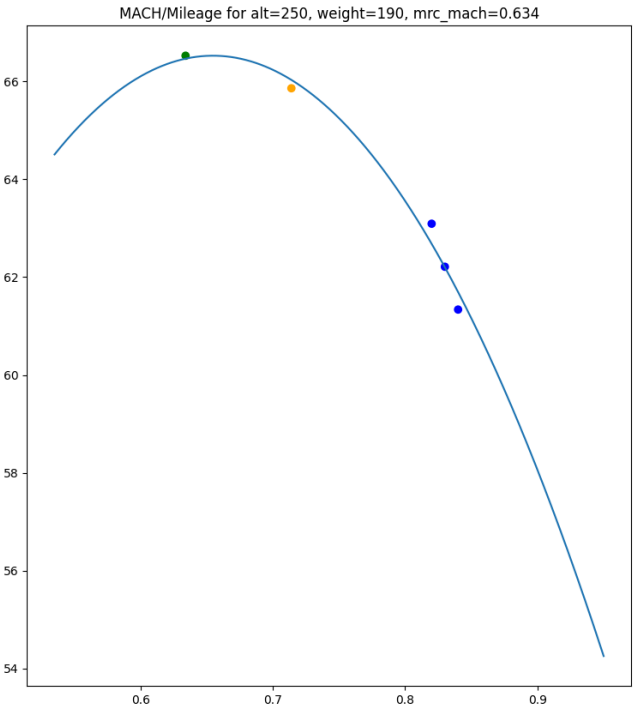
Found: 0.612 MRC and 0.694 LRC

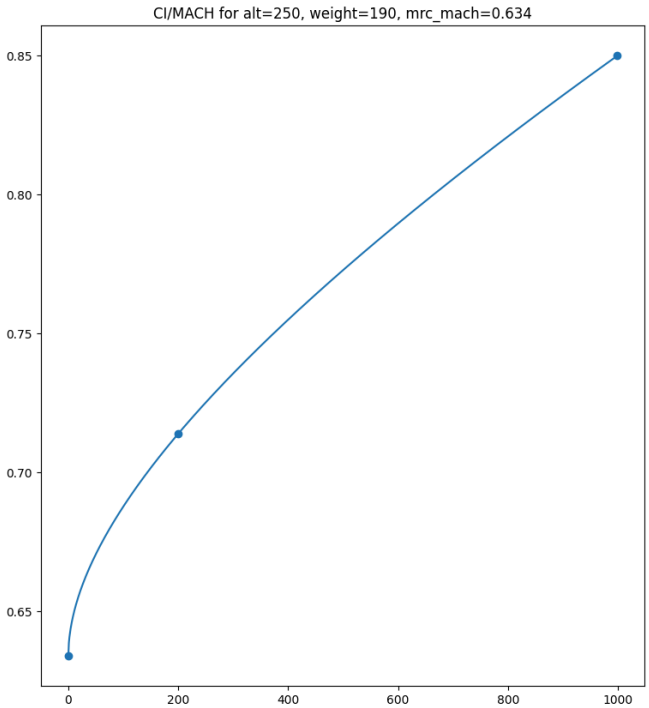




Sanity check: 0.612 MRC and 0.694 LRC

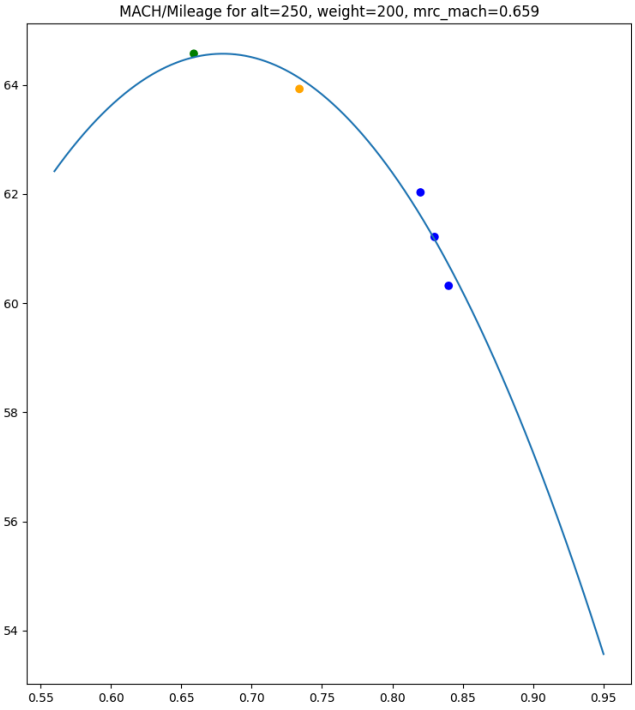
Found: 0.634 MRC and 0.714 LRC

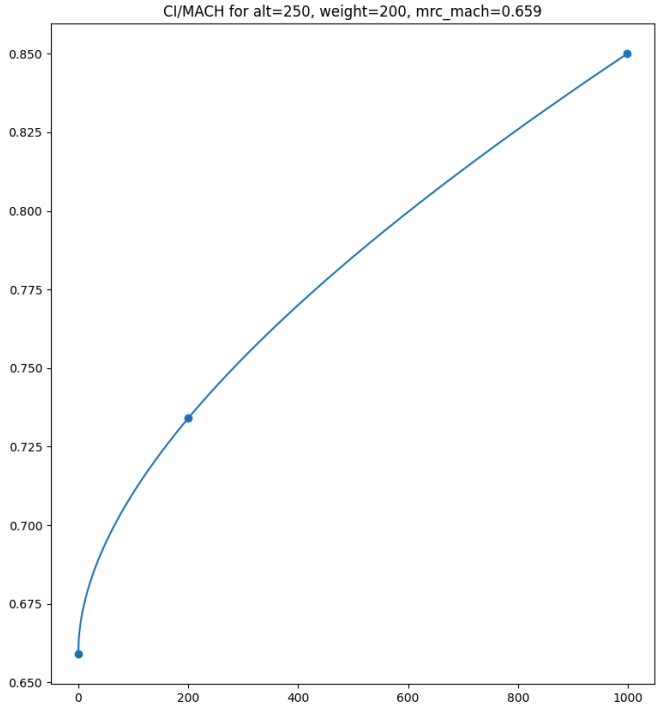




Sanity check: 0.634 MRC and 0.714 LRC

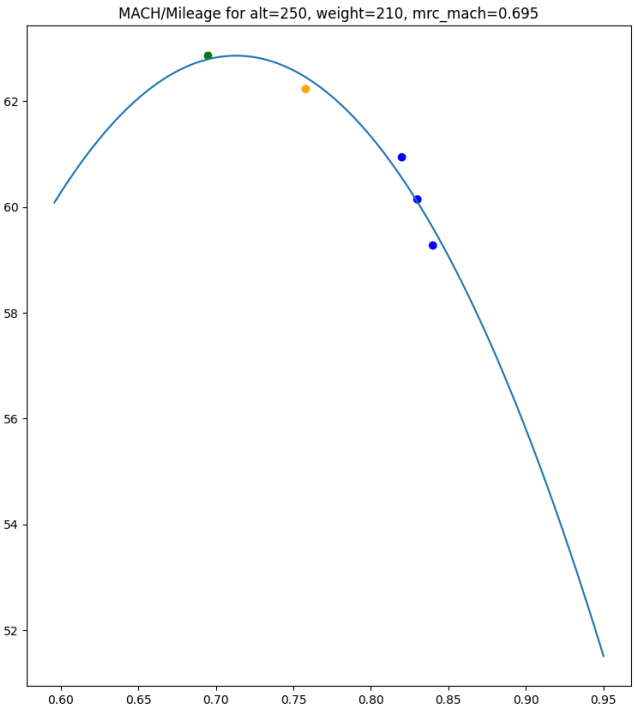
Found: 0.659 MRC and 0.734 LRC

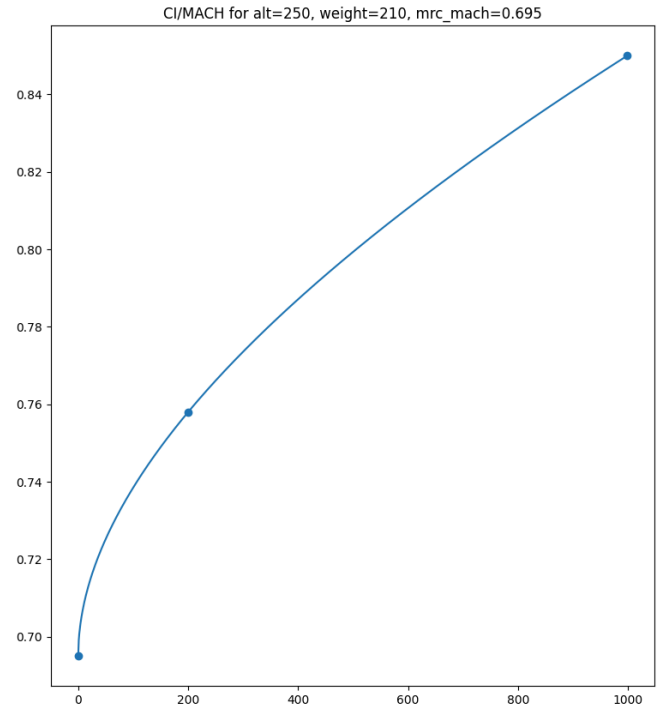




Sanity check: 0.659 MRC and 0.734 LRC

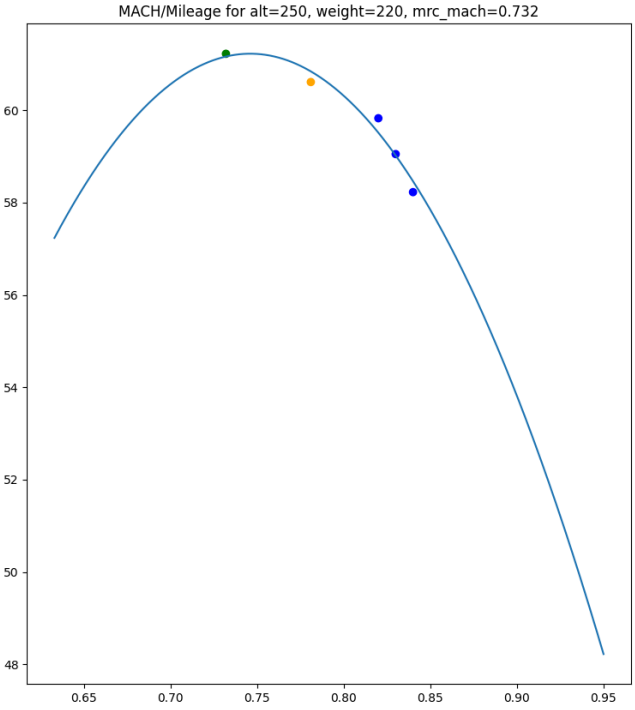
Found: 0.695 MRC and 0.758 LRC

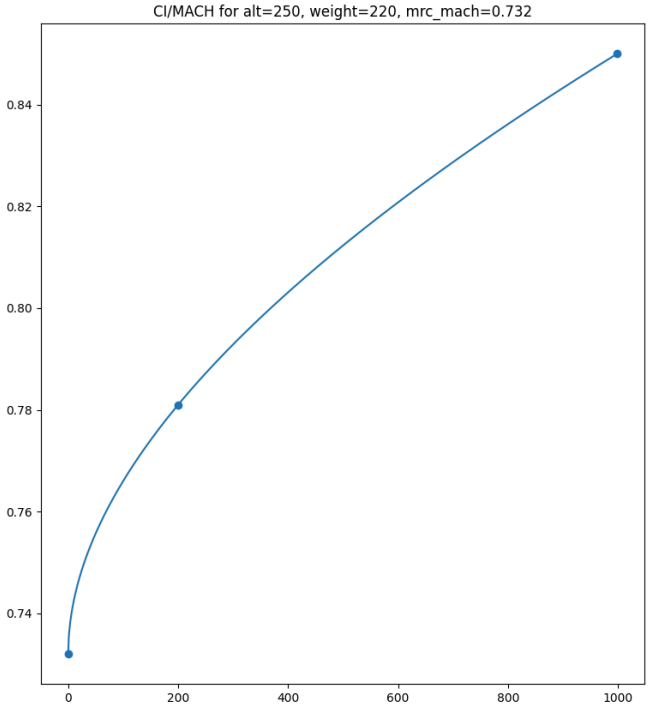




Sanity check: 0.695 MRC and 0.758 LRC

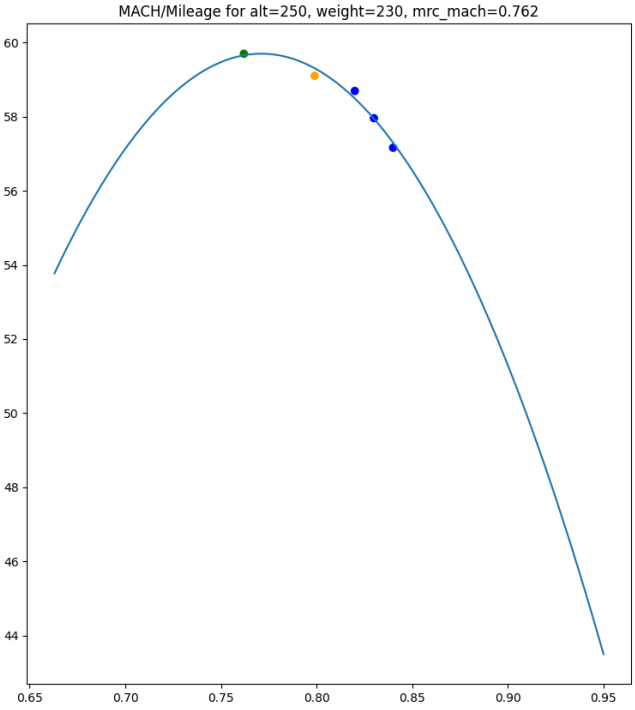
Found: 0.732 MRC and 0.781 LRC

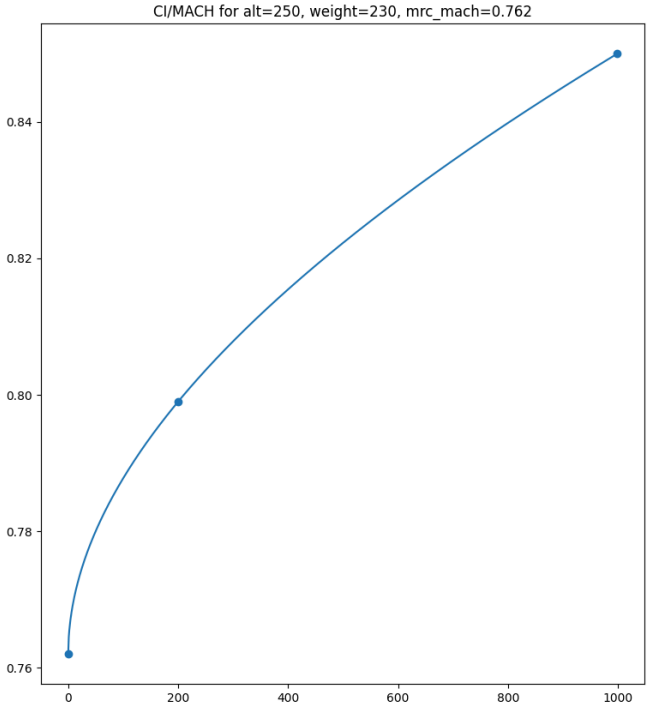




Sanity check: 0.732 MRC and 0.781 LRC

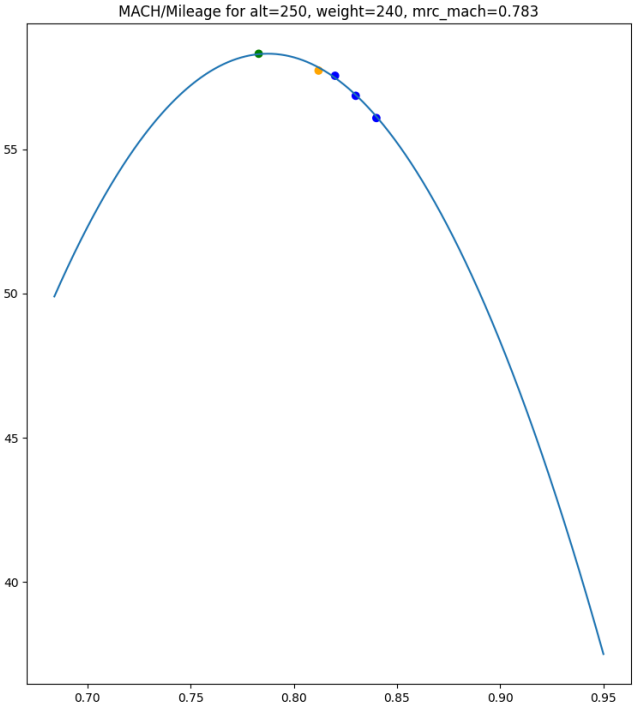
Found: 0.762 MRC and 0.799 LRC

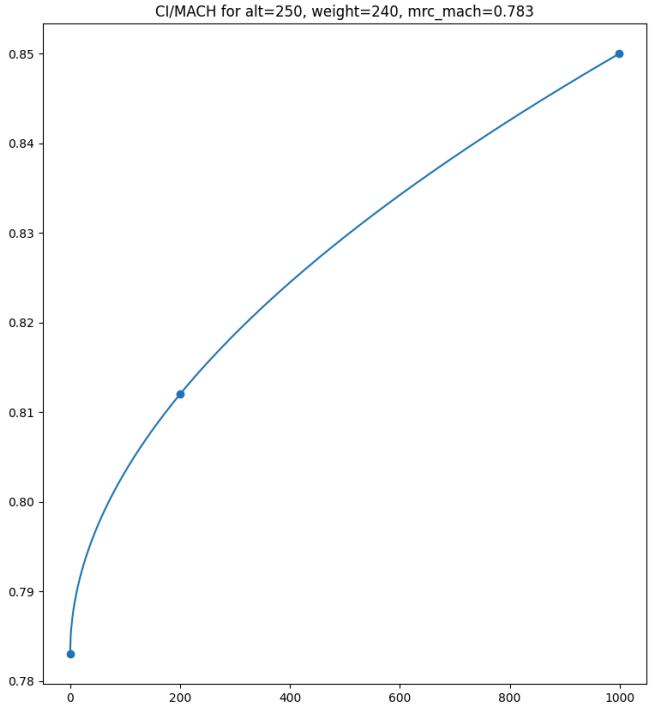




Sanity check: 0.762 MRC and 0.799 LRC

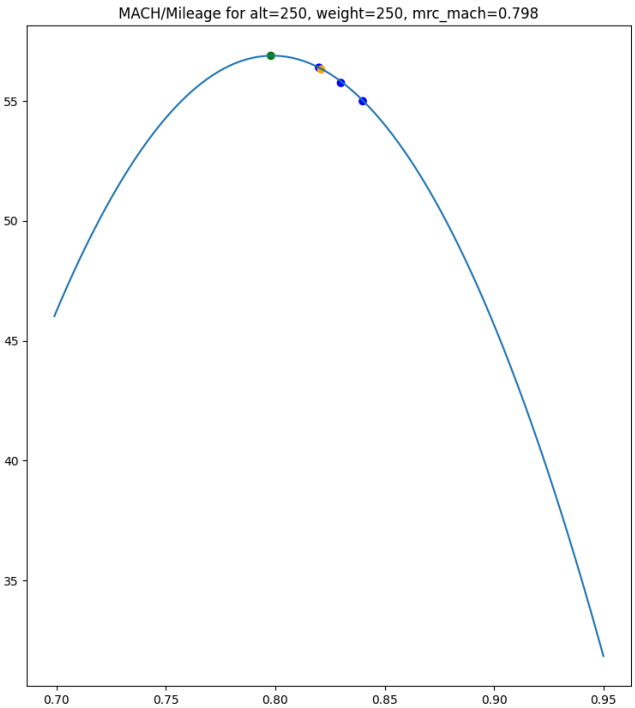
Found: 0.783 MRC and 0.812 LRC

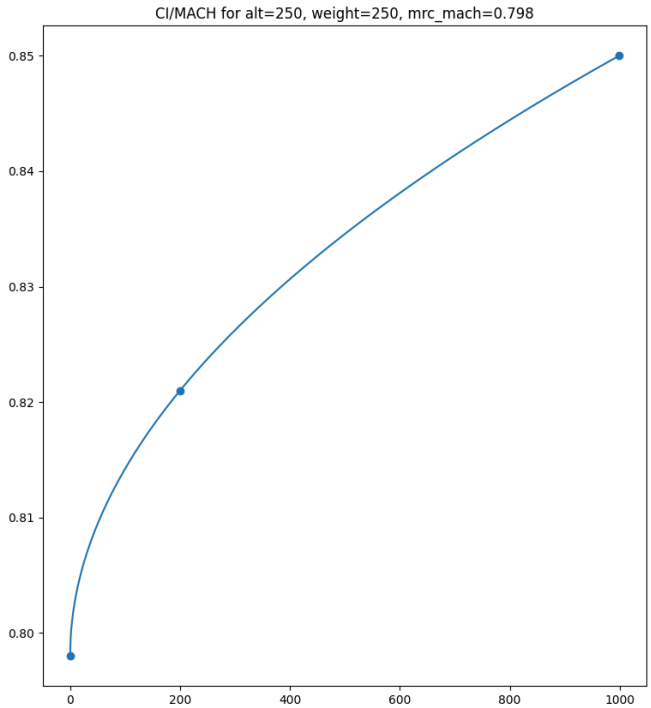




Sanity check: 0.783 MRC and 0.812 LRC

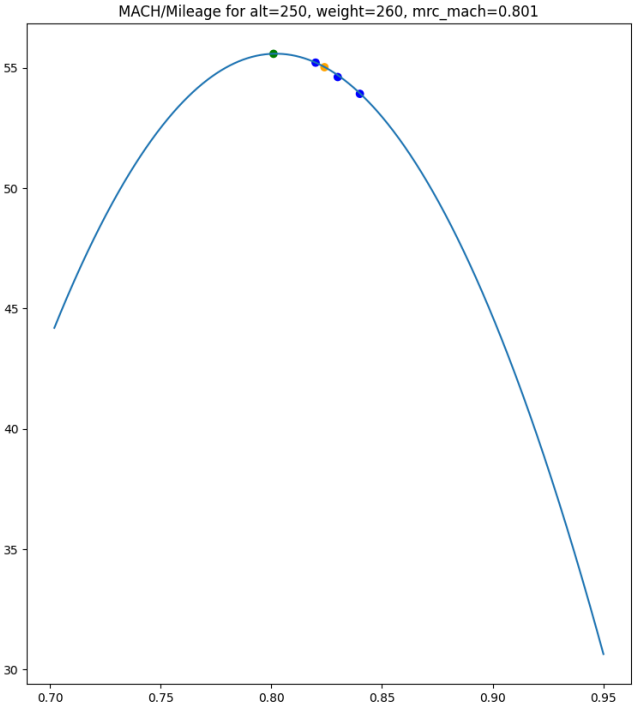
Found: 0.798 MRC and 0.821 LRC

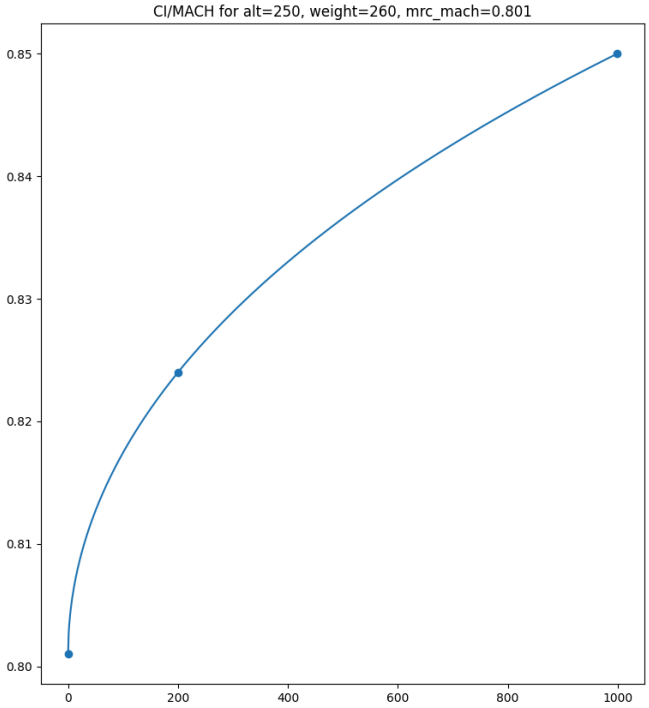




Sanity check: 0.798 MRC and 0.821 LRC

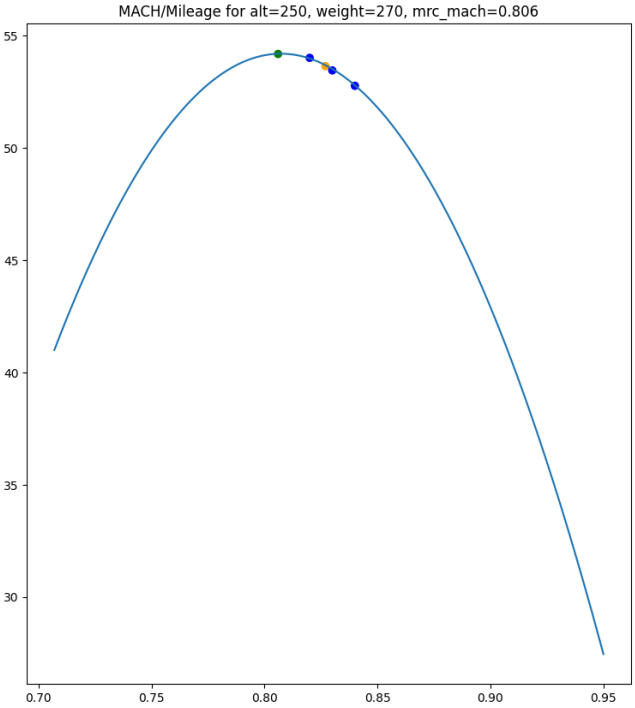
Found: 0.801 MRC and 0.824 LRC

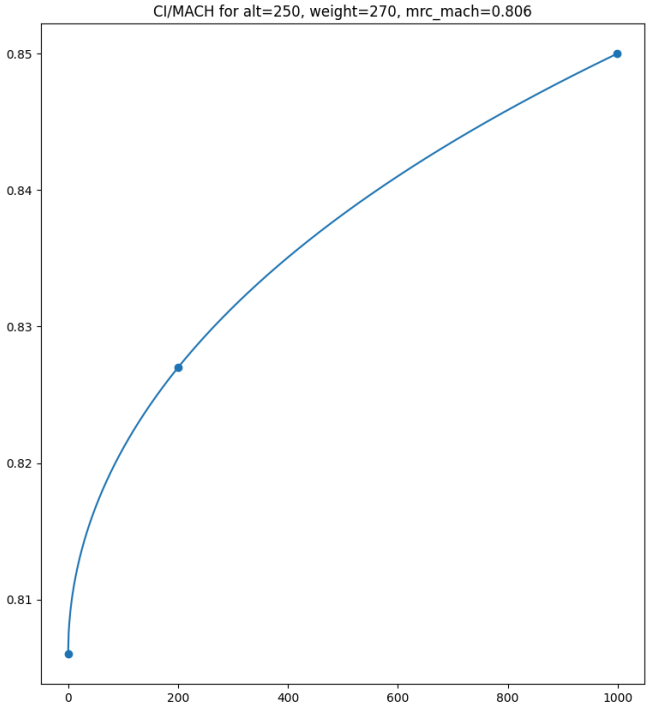




Sanity check: 0.801 MRC and 0.824 LRC

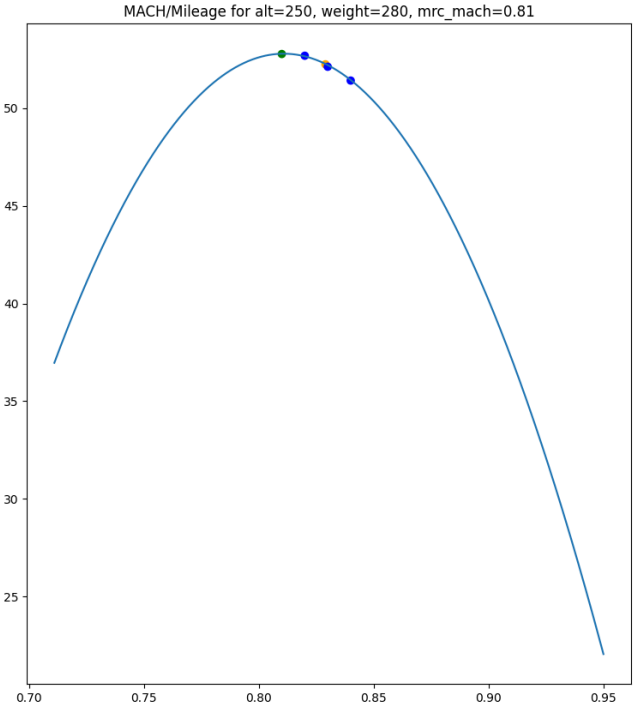
Found: 0.806 MRC and 0.827 LRC

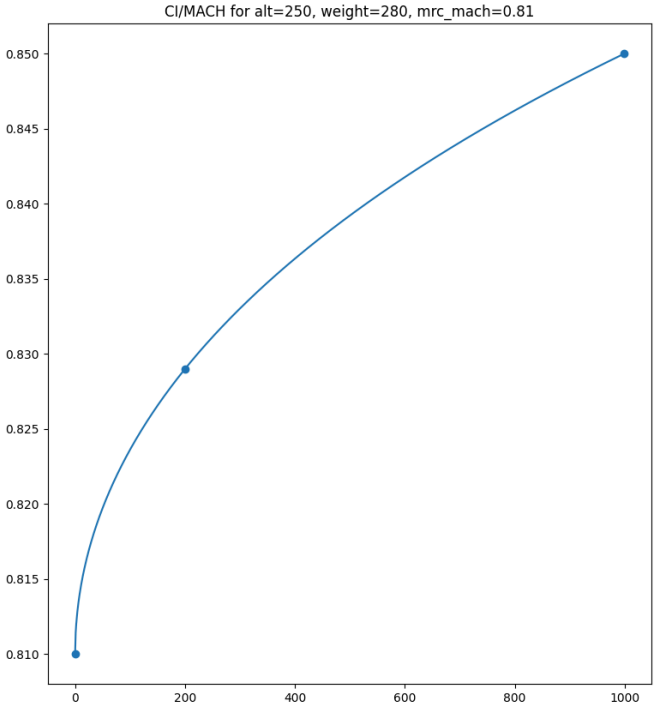




Sanity check: 0.806 MRC and 0.827 LRC

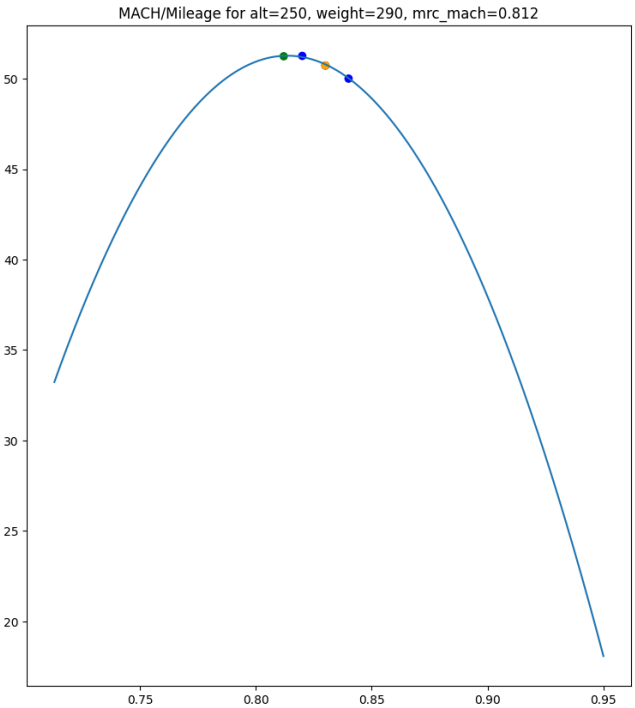
Found: 0.81 MRC and 0.829 LRC

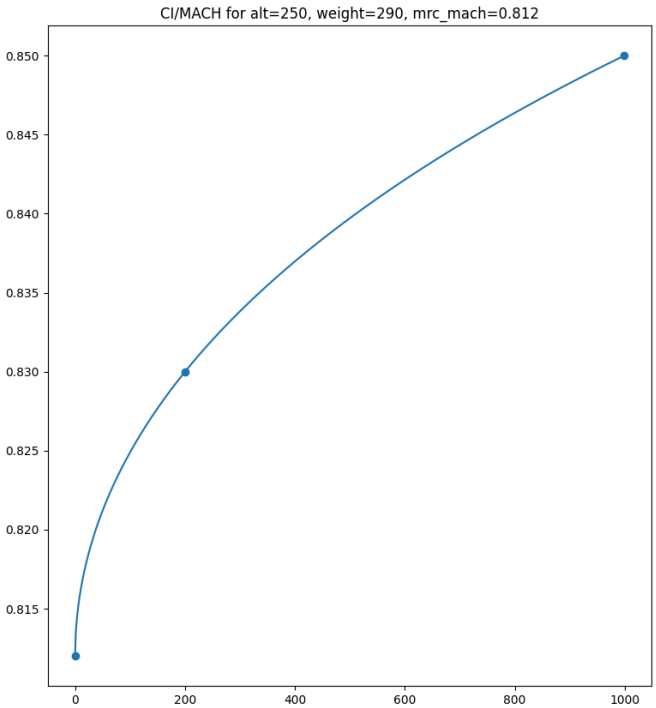




Sanity check: 0.81 MRC and 0.829 LRC

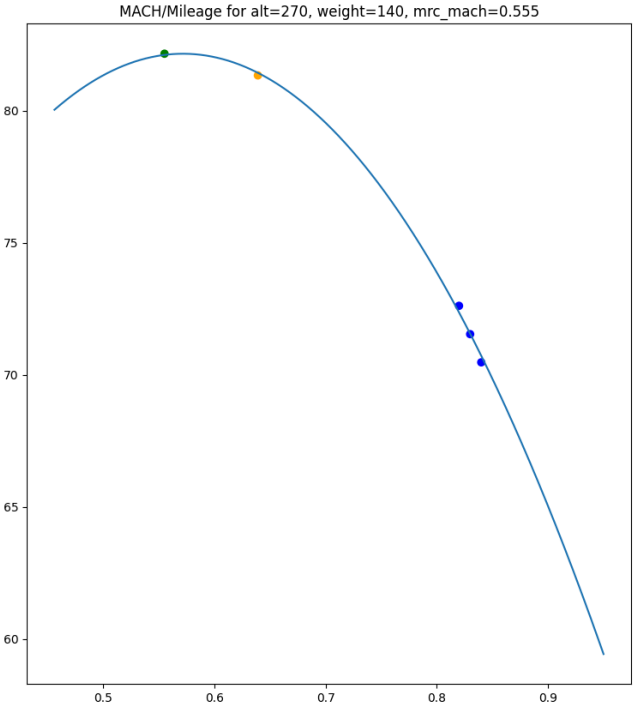
Found: 0.812 MRC and 0.83 LRC

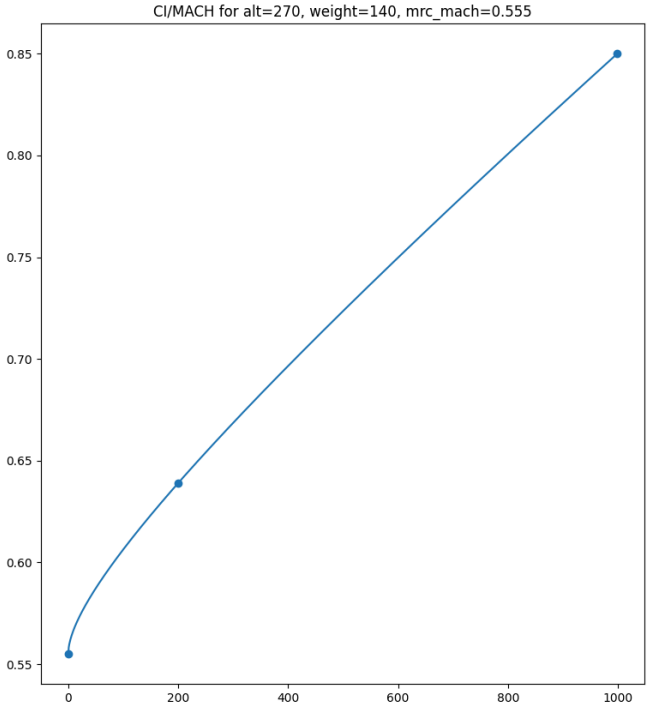




Sanity check: 0.812 MRC and 0.83 LRC

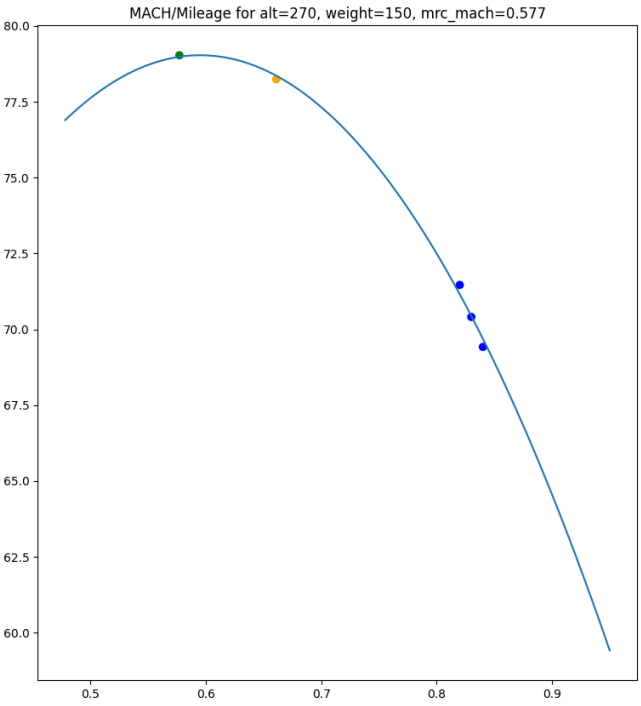
Found: 0.555 MRC and 0.639 LRC

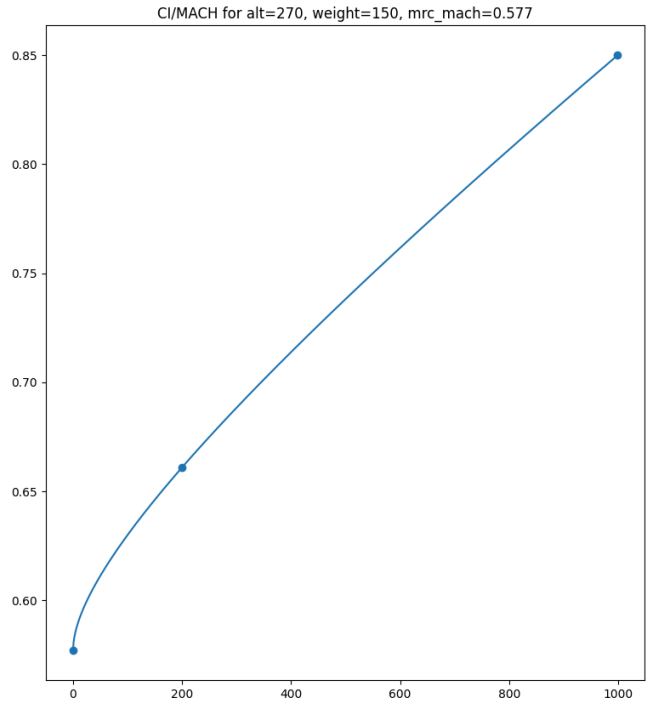




Sanity check: 0.555 MRC and 0.639 LRC

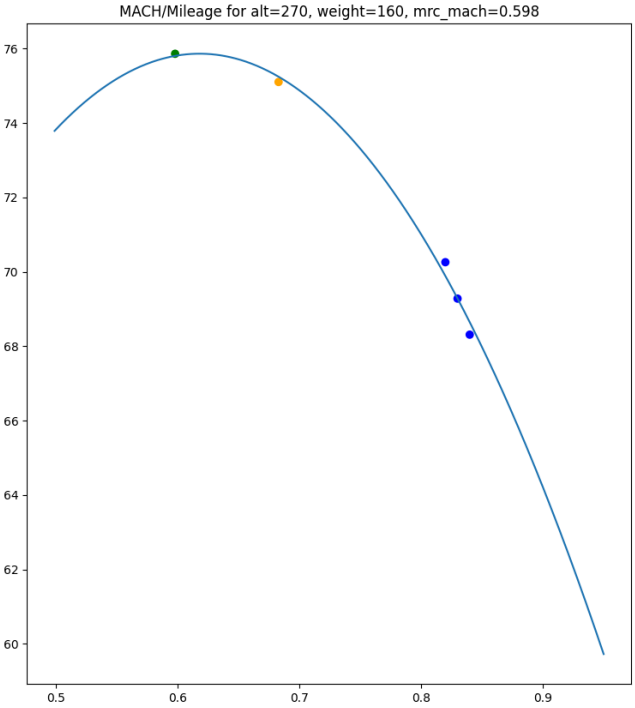
Found: 0.577 MRC and 0.661 LRC

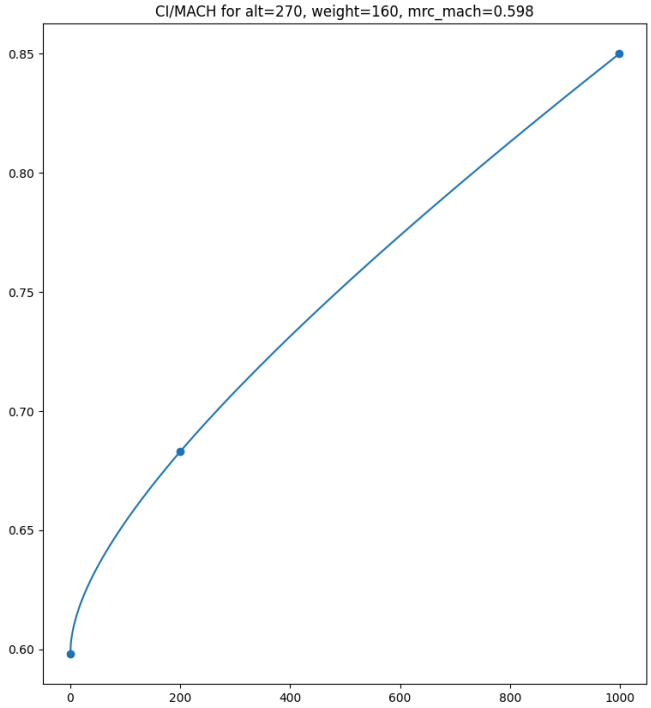




Sanity check: 0.577 MRC and 0.661 LRC

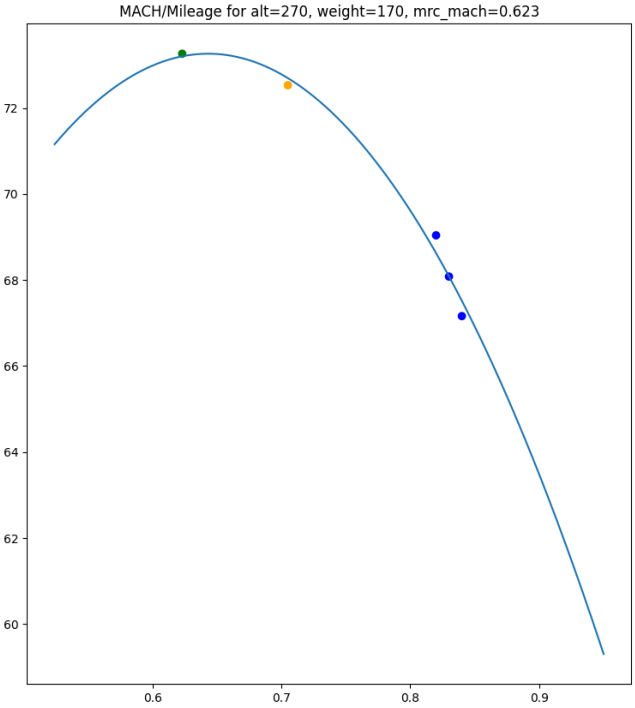
Found: 0.598 MRC and 0.683 LRC

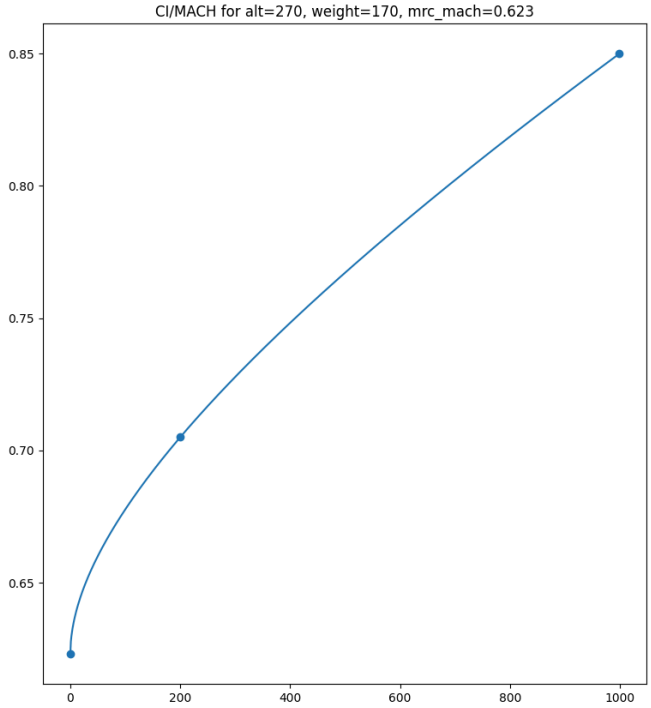




Sanity check: 0.598 MRC and 0.683 LRC

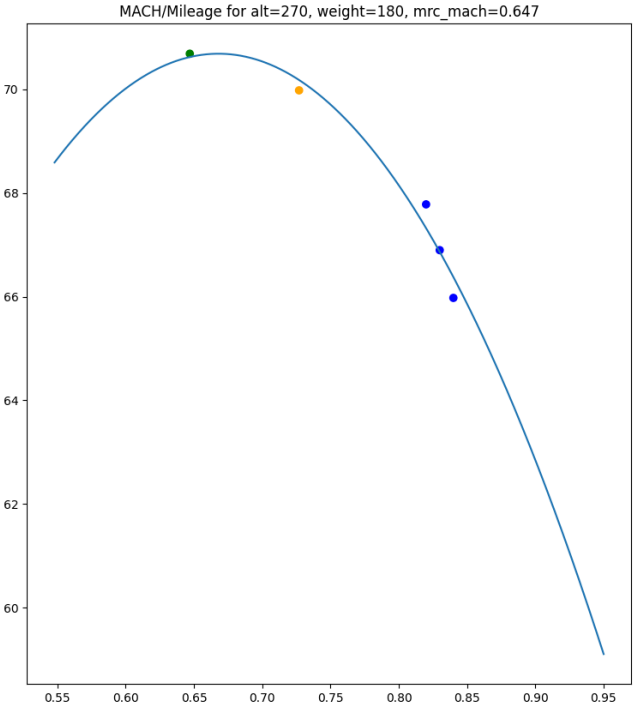
Found: 0.623 MRC and 0.705 LRC

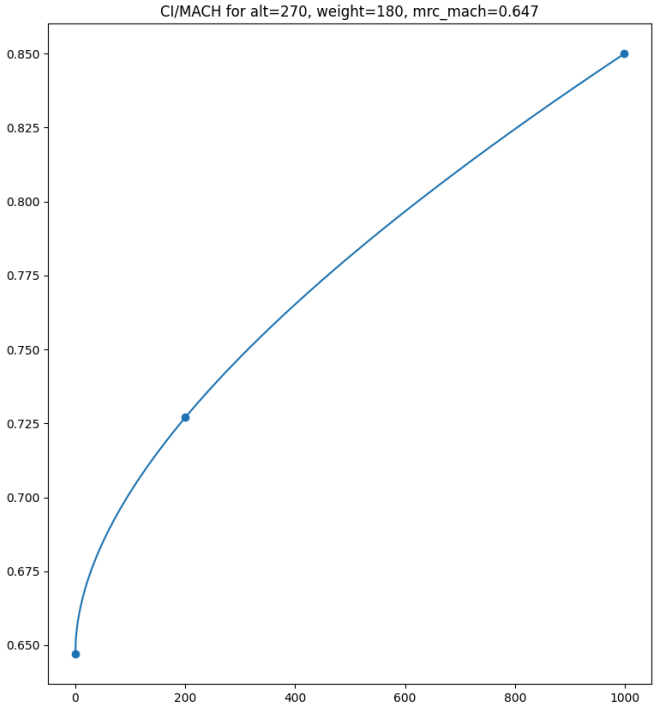




Sanity check: 0.623 MRC and 0.705 LRC

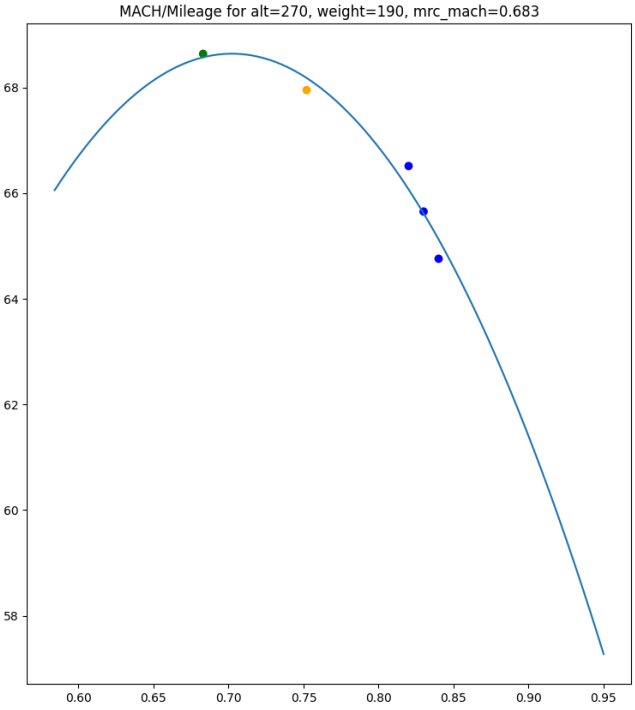
Found: 0.647 MRC and 0.727 LRC

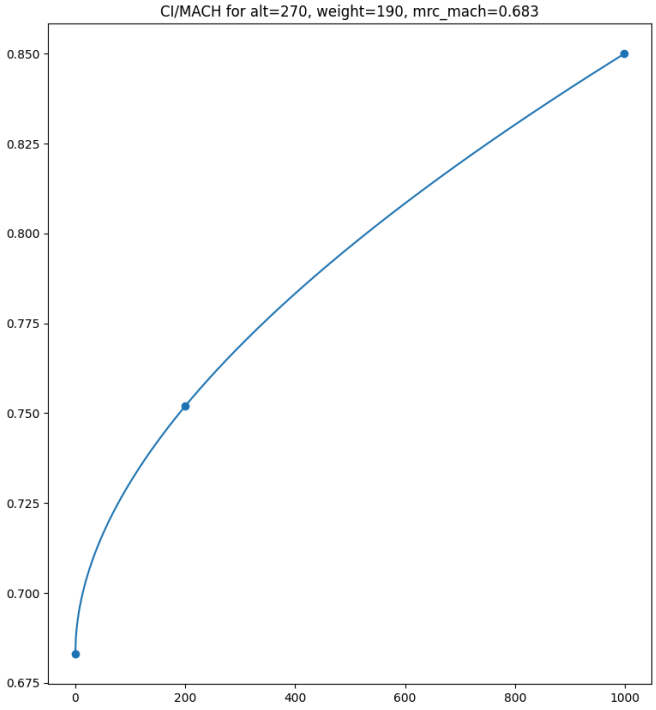




Sanity check: 0.647 MRC and 0.727 LRC

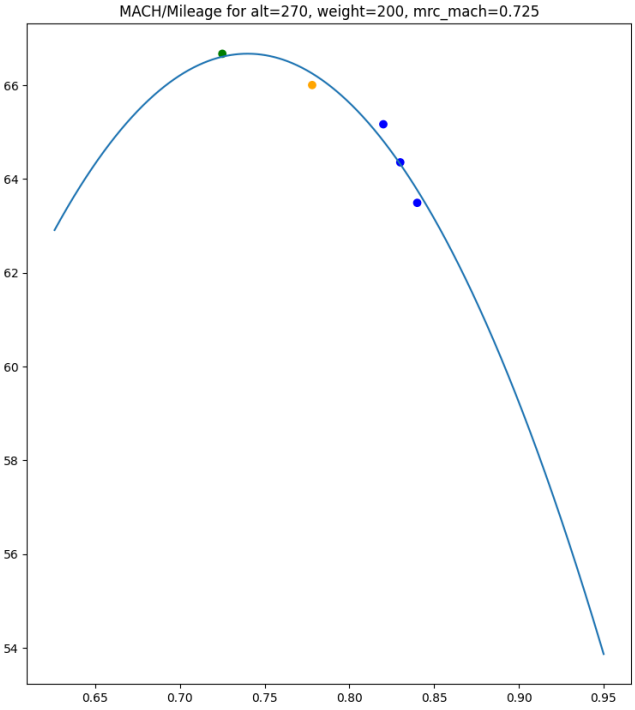
Found: 0.683 MRC and 0.752 LRC

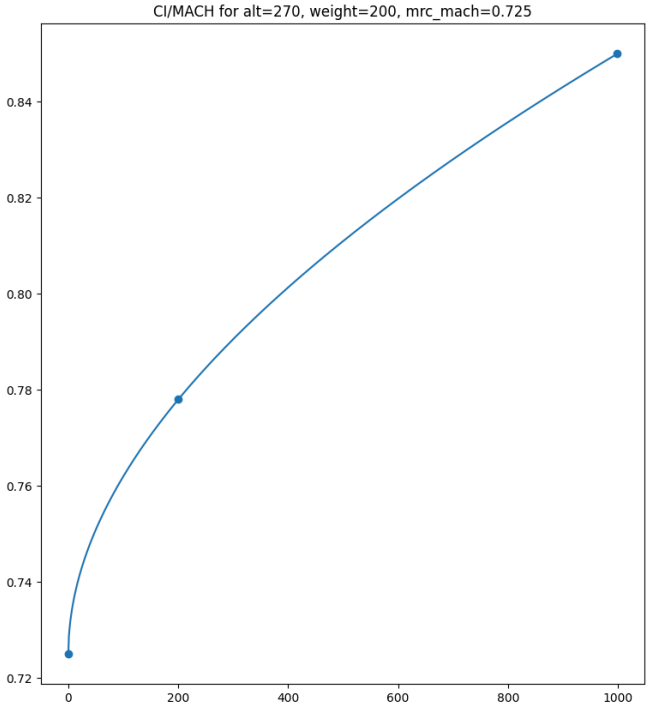




Sanity check: 0.683 MRC and 0.752 LRC

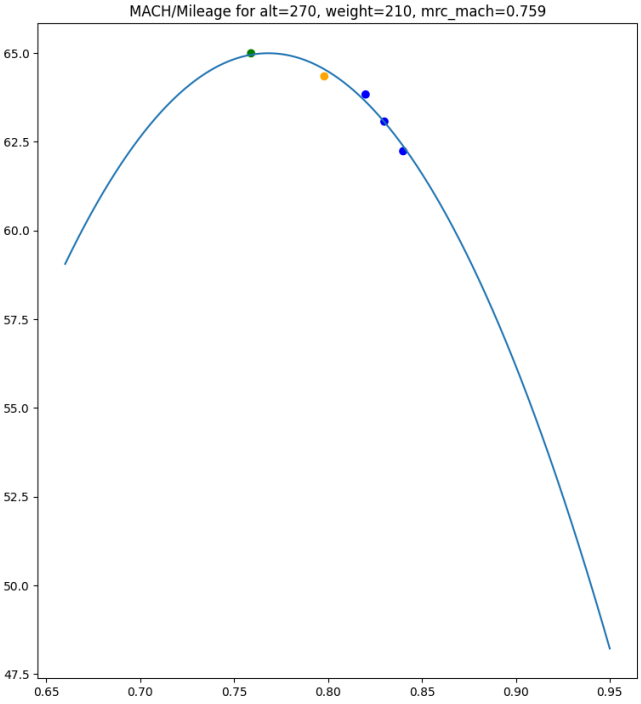
Found: 0.725 MRC and 0.778 LRC

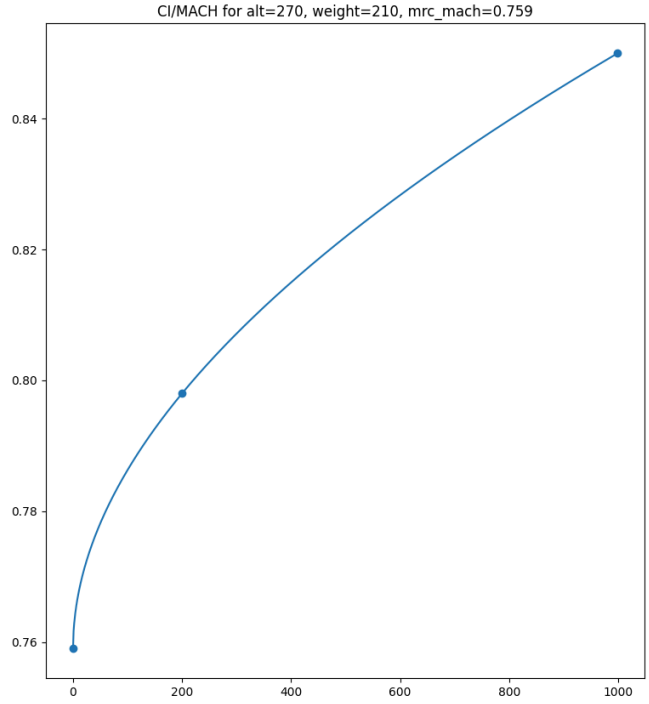




Sanity check: 0.725 MRC and 0.778 LRC

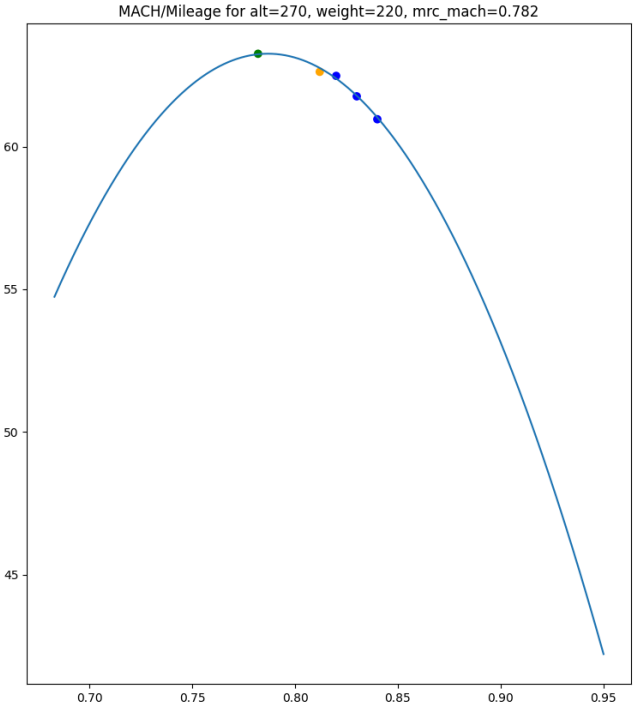
Found: 0.759 MRC and 0.798 LRC

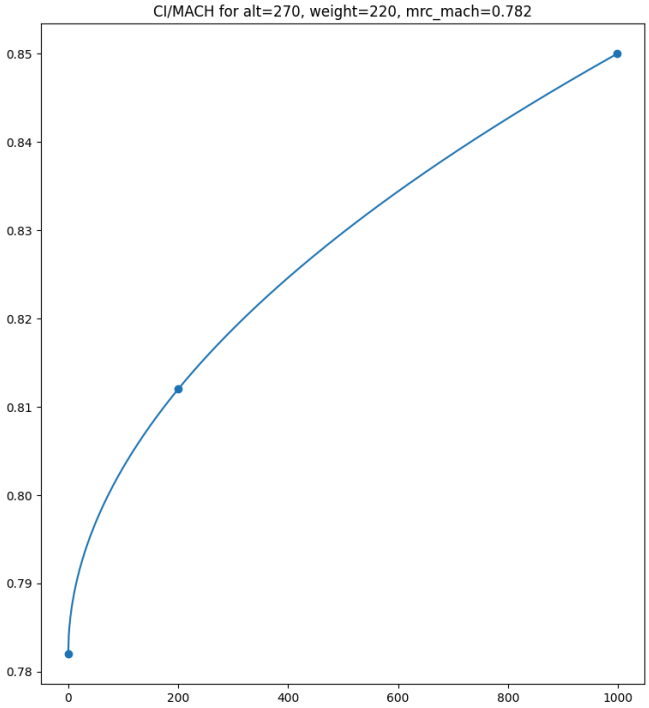




Sanity check: 0.759 MRC and 0.798 LRC

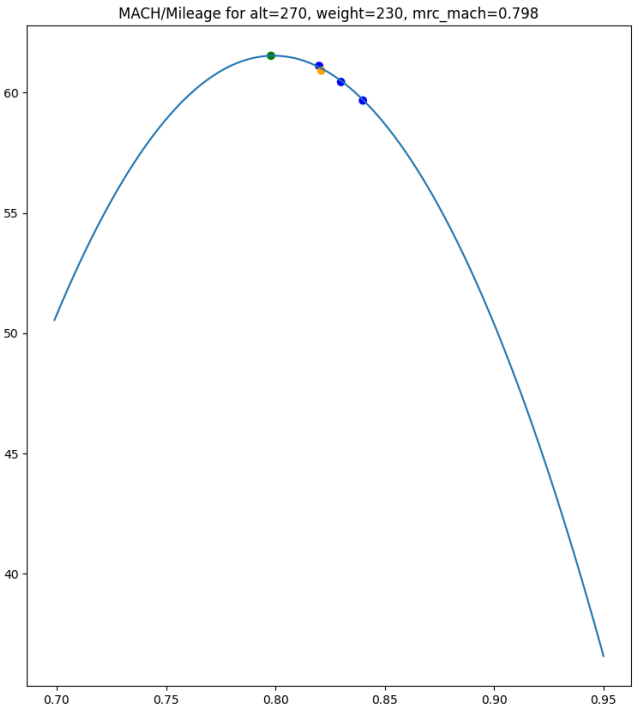
Found: 0.782 MRC and 0.812 LRC

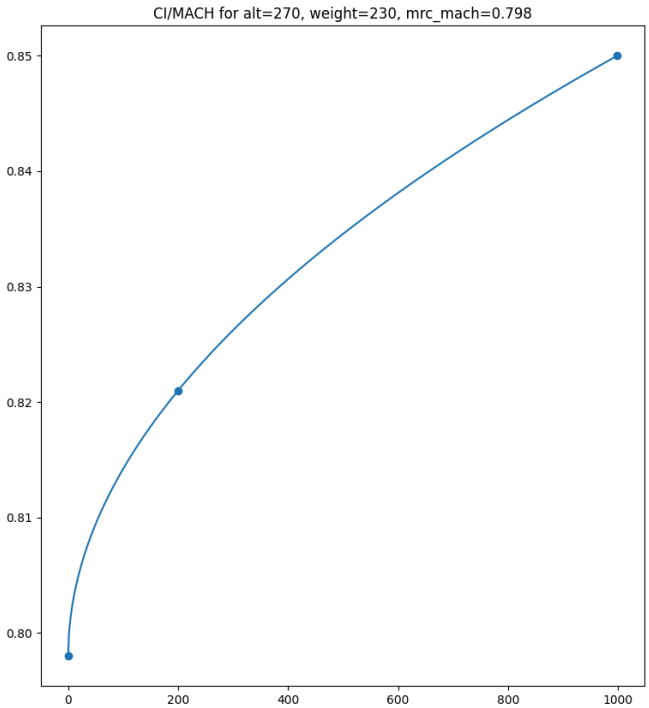




Sanity check: 0.782 MRC and 0.812 LRC

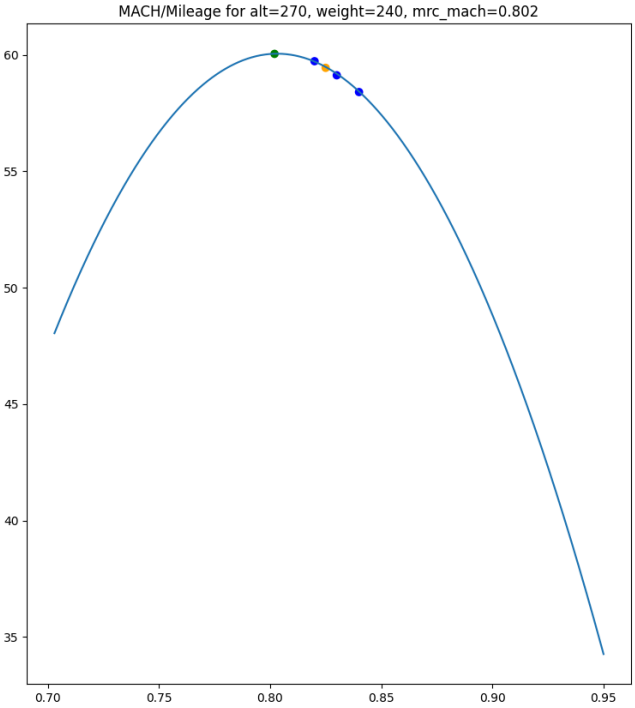
Found: 0.798 MRC and 0.821 LRC

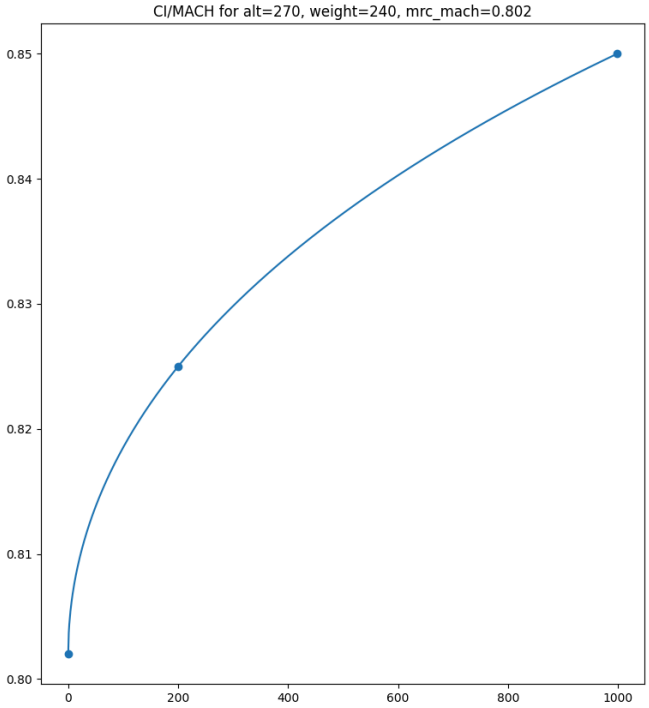




Sanity check: 0.798 MRC and 0.821 LRC

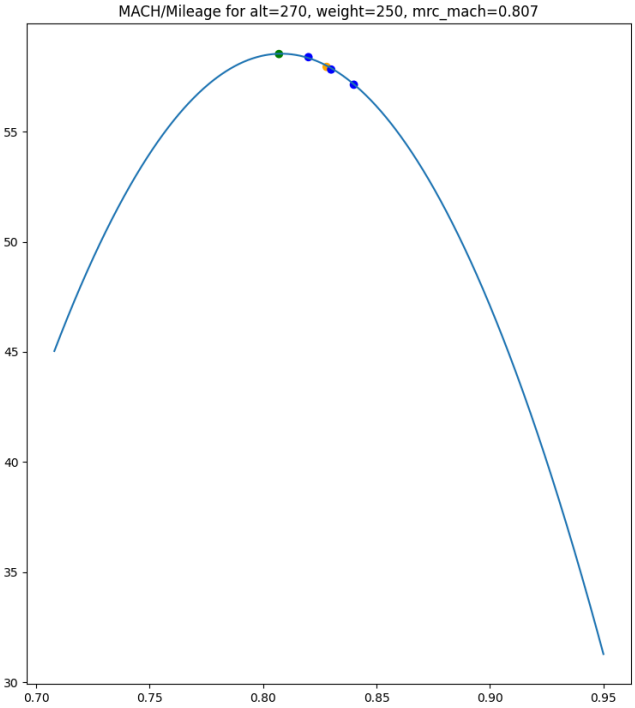
Found: 0.802 MRC and 0.825 LRC

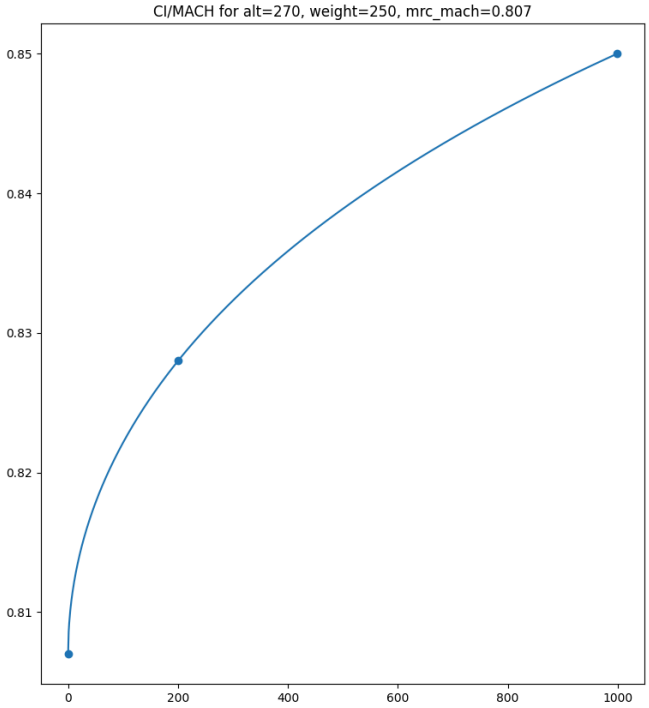




Sanity check: 0.802 MRC and 0.825 LRC

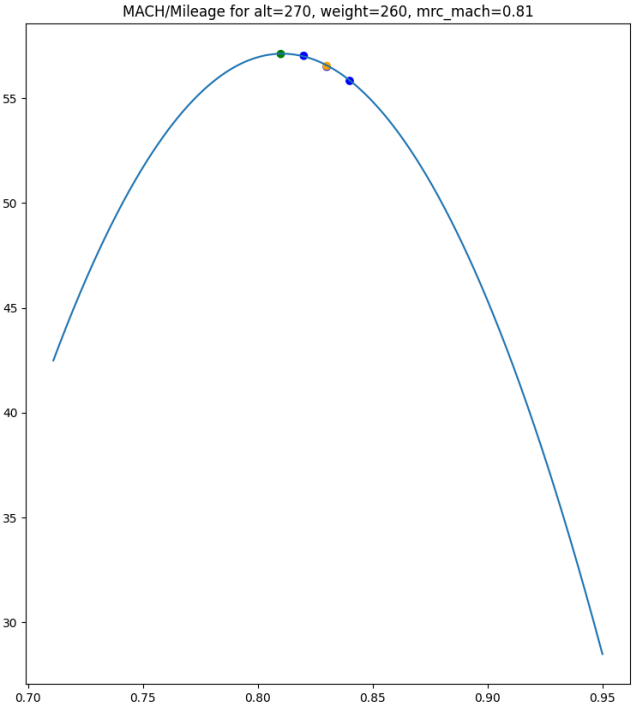
Found: 0.807 MRC and 0.828 LRC

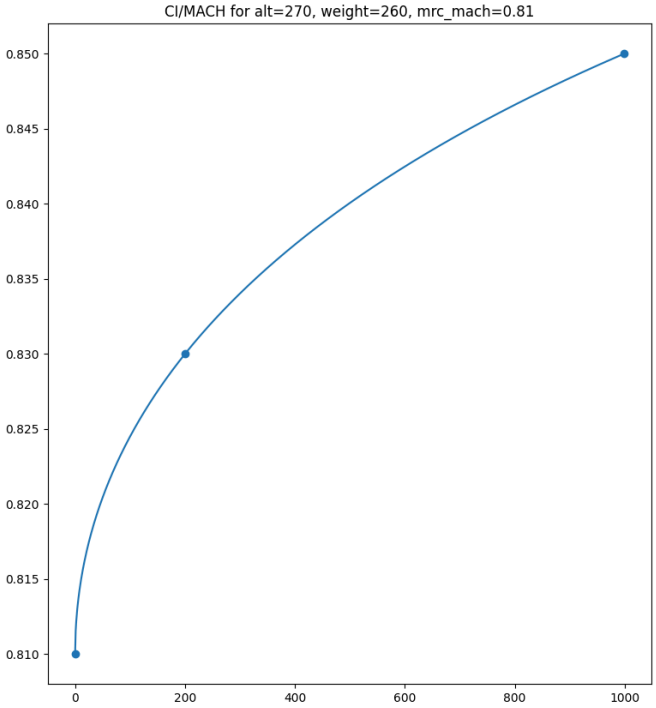




Sanity check: 0.807 MRC and 0.828 LRC

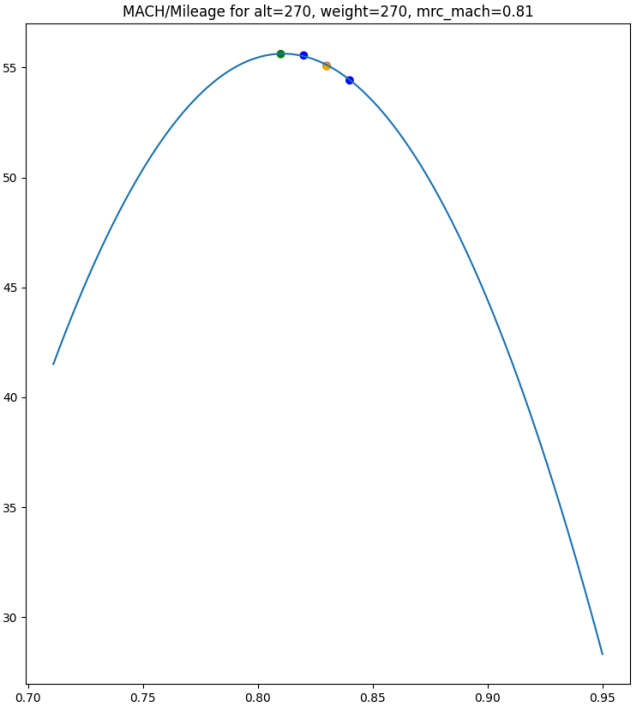
Found: 0.81 MRC and 0.83 LRC

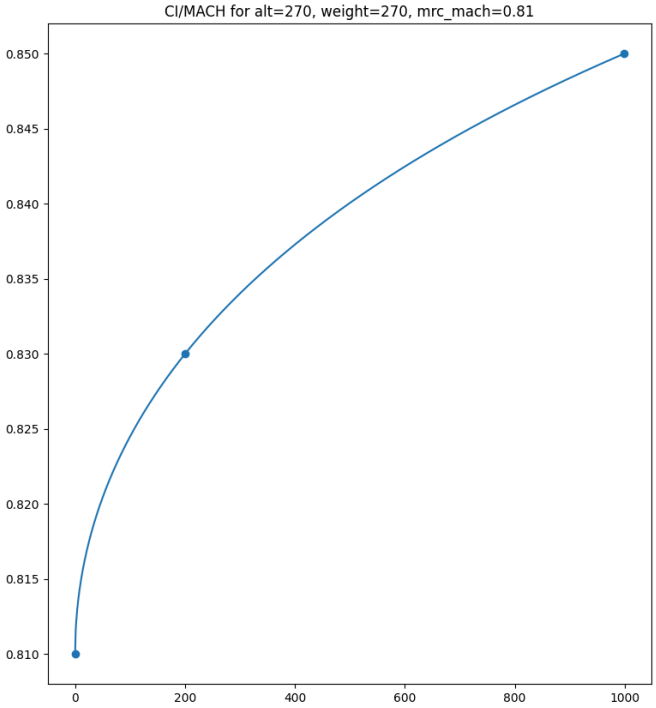




Sanity check: 0.81 MRC and 0.83 LRC

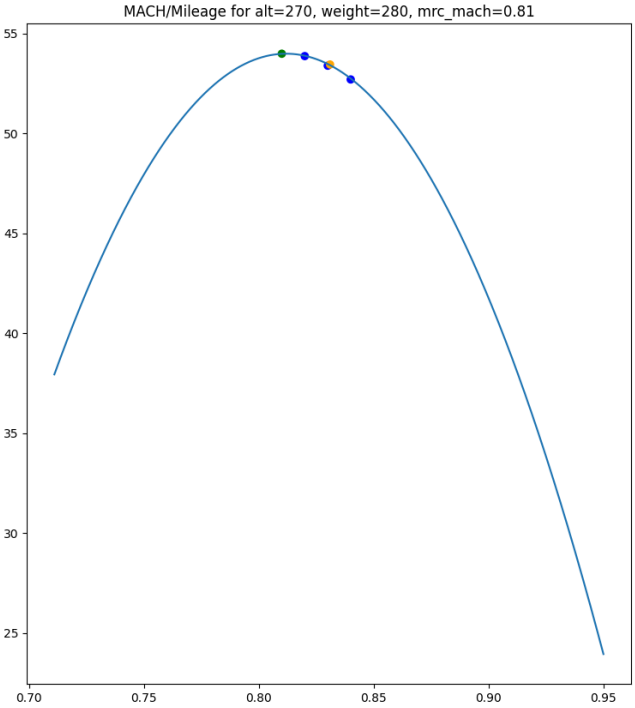
Found: 0.81 MRC and 0.83 LRC

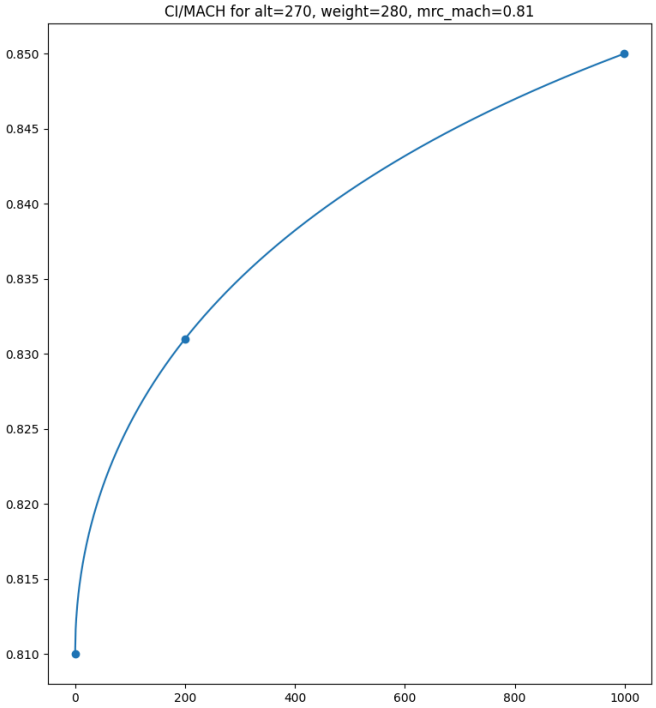




Sanity check: 0.81 MRC and 0.83 LRC

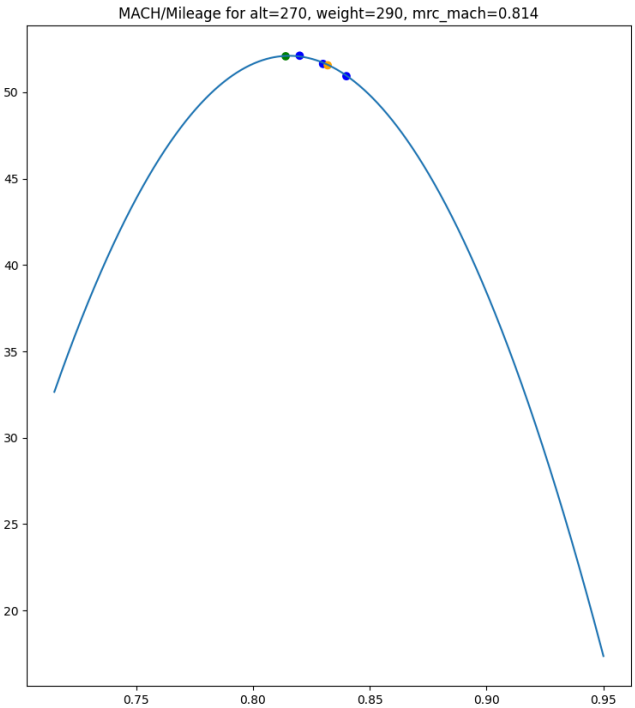
Found: 0.81 MRC and 0.831 LRC

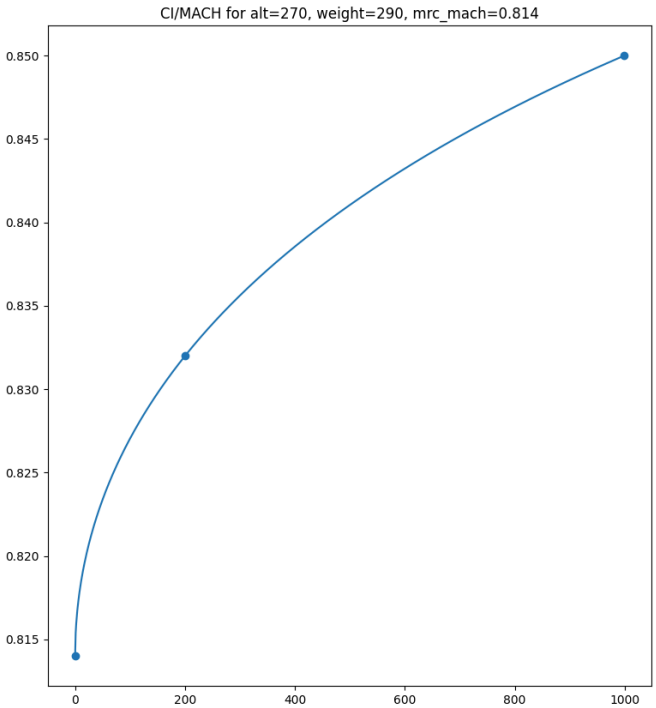




Sanity check: 0.81 MRC and 0.831 LRC

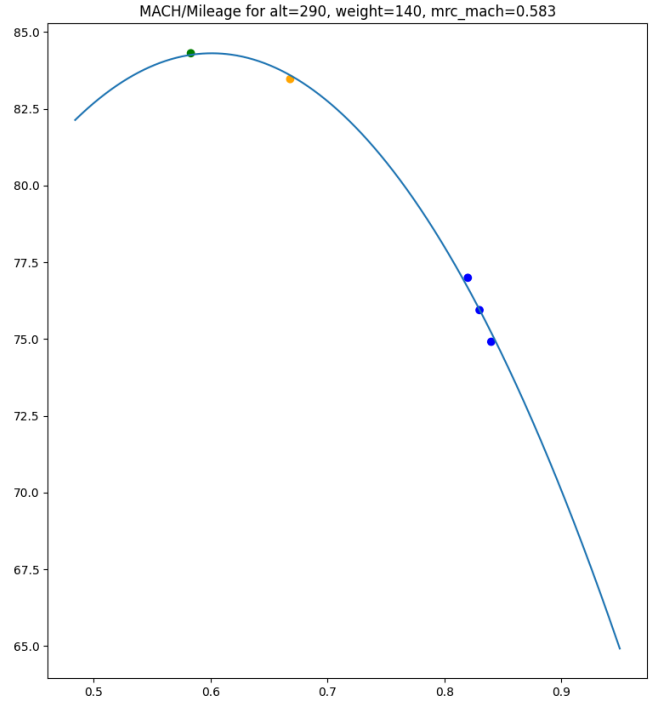
Found: 0.814 MRC and 0.832 LRC

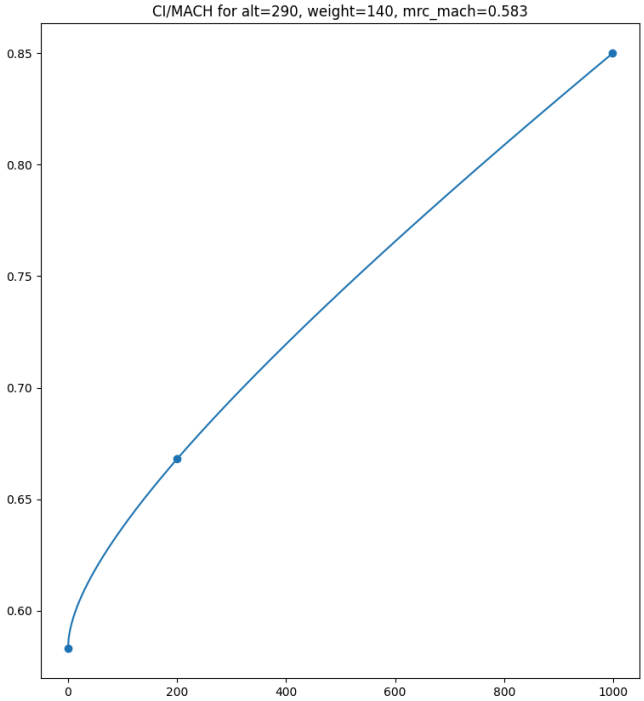




Sanity check: 0.814 MRC and 0.832 LRC

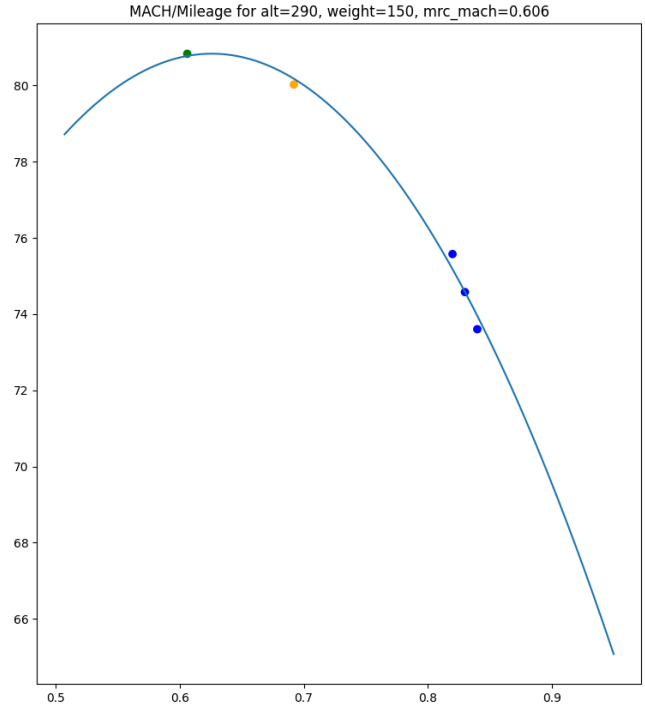
Found: 0.583 MRC and 0.668 LRC

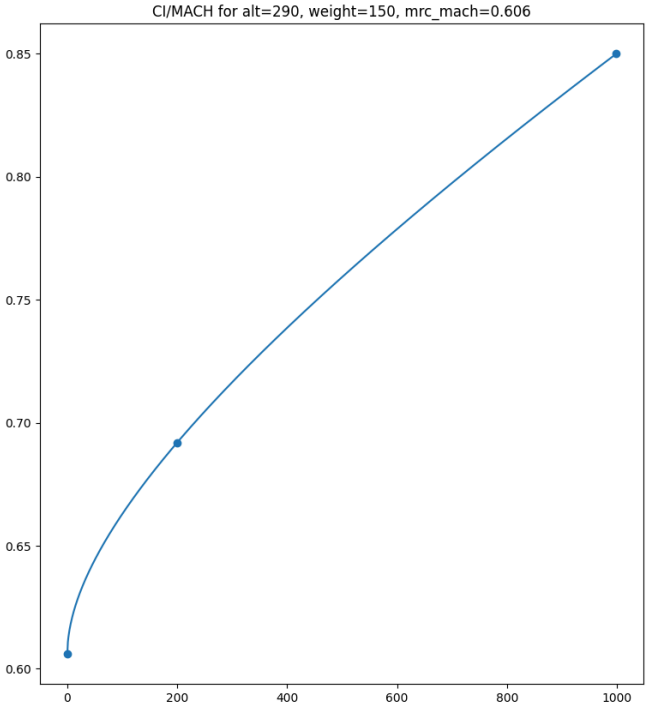




Sanity check: 0.583 MRC and 0.668 LRC

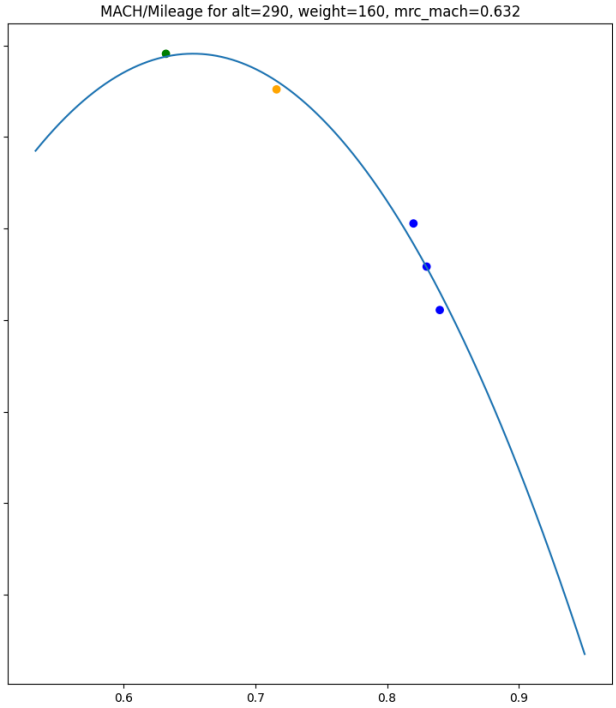
Found: 0.606 MRC and 0.692 LRC

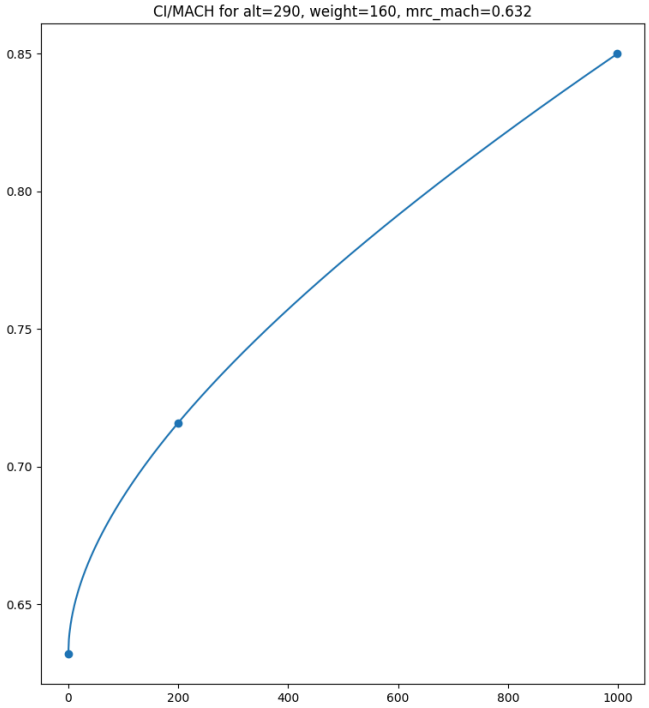




Sanity check: 0.606 MRC and 0.692 LRC

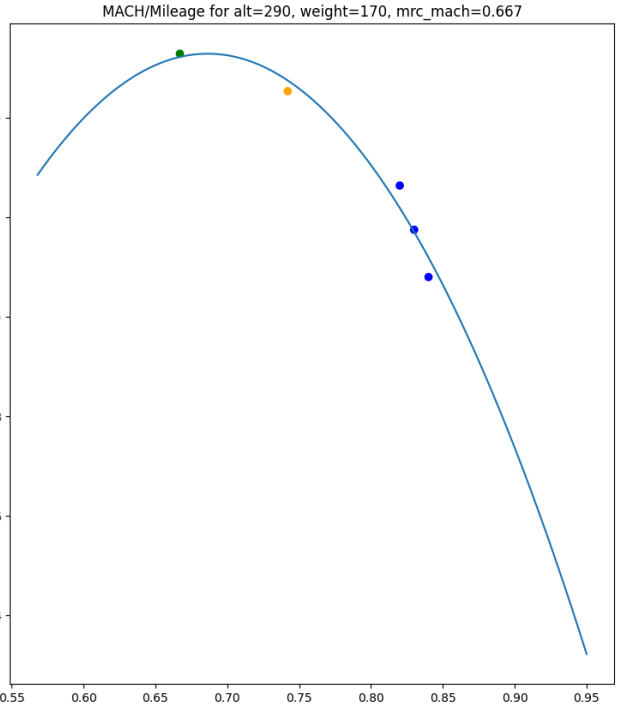
Found: 0.632 MRC and 0.716 LRC

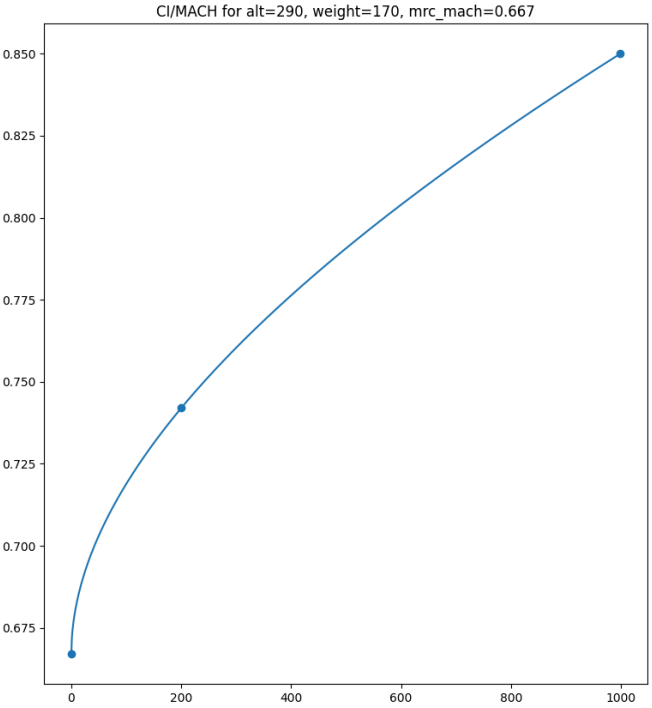




Sanity check: 0.632 MRC and 0.716 LRC

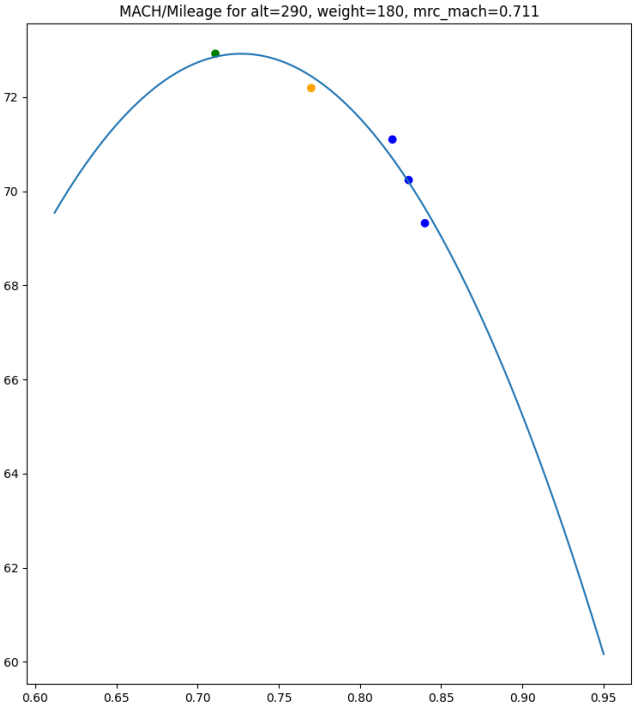
Found: 0.667 MRC and 0.742 LRC

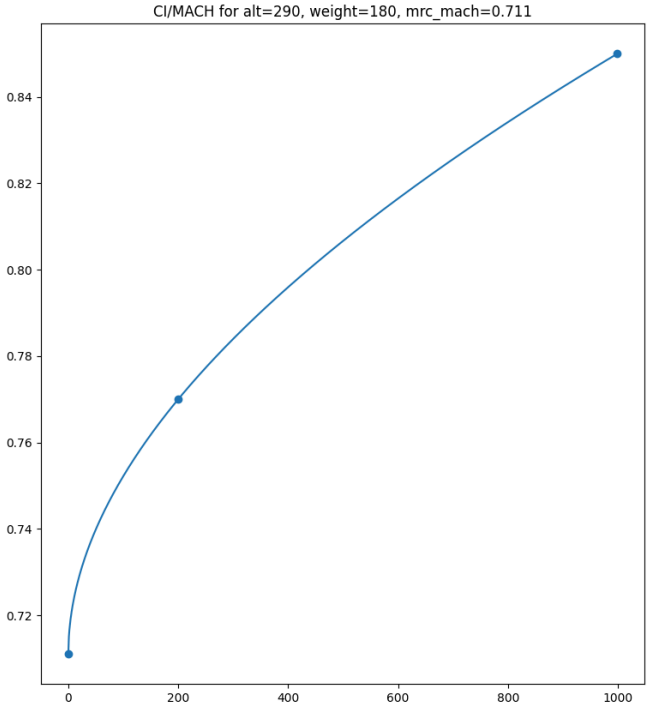




Sanity check: 0.667 MRC and 0.742 LRC

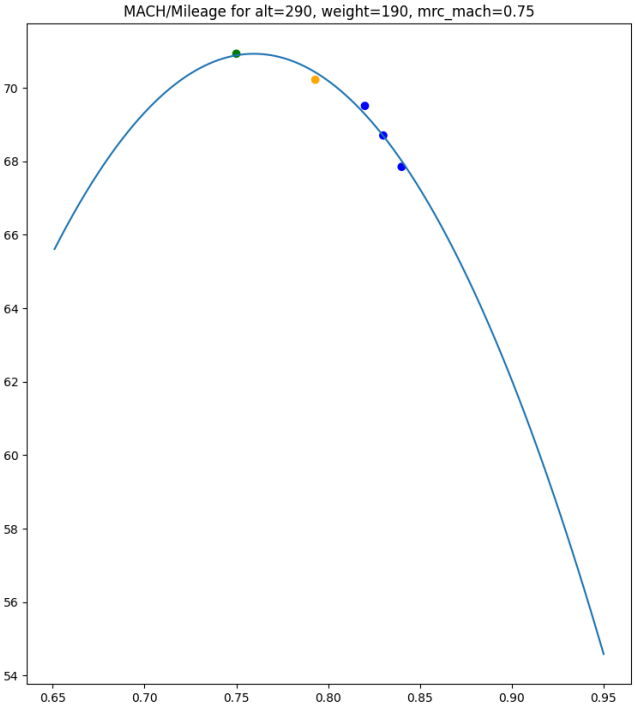
Found: 0.711 MRC and 0.77 LRC

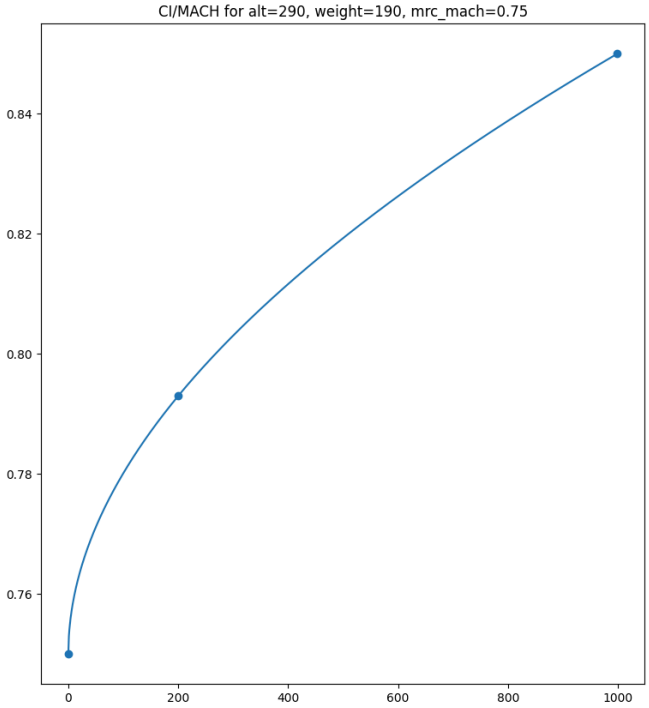




Sanity check: 0.711 MRC and 0.77 LRC

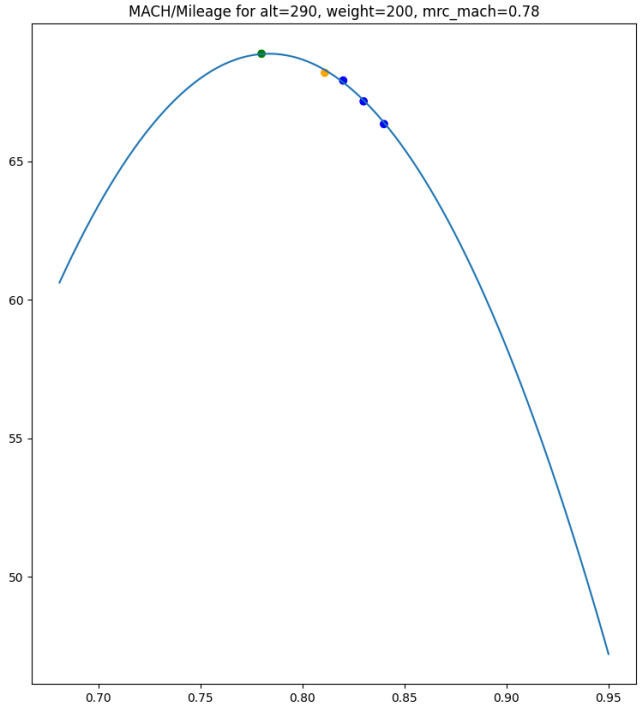
Found: 0.75 MRC and 0.793 LRC

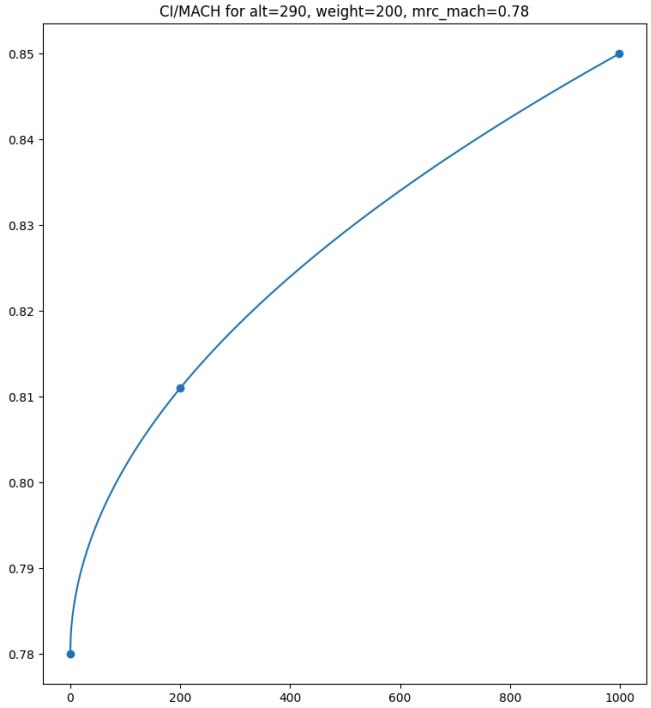




Sanity check: 0.75 MRC and 0.793 LRC

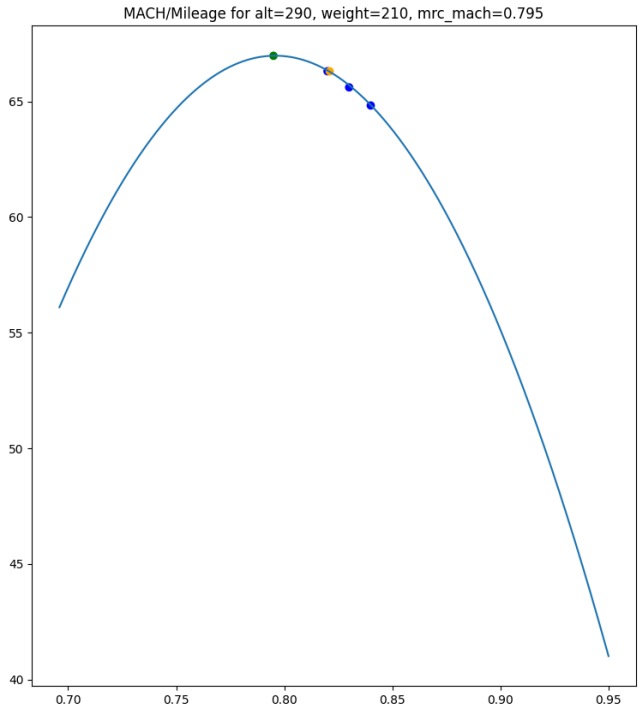
Found: 0.78 MRC and 0.811 LRC

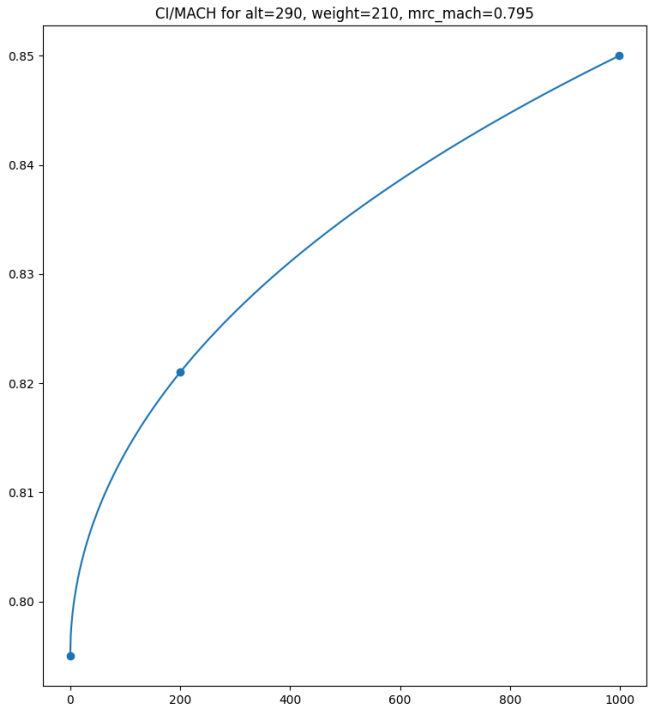




Sanity check: 0.78 MRC and 0.811 LRC

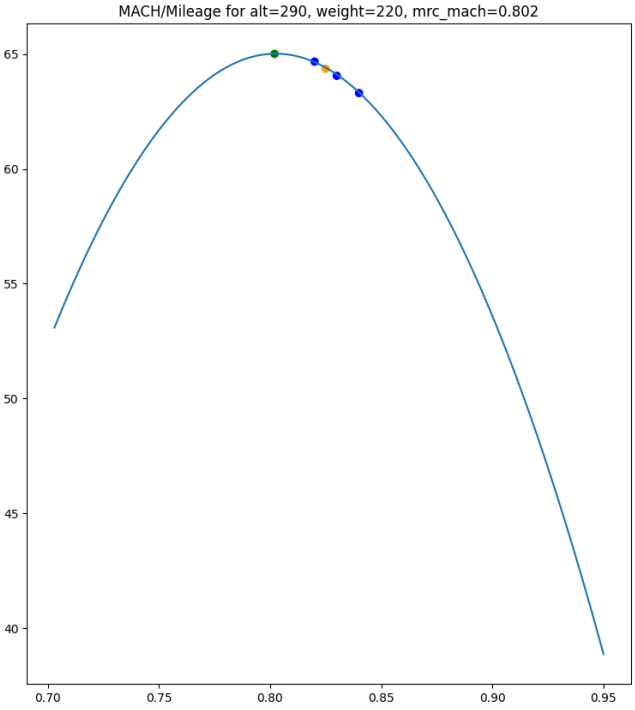
Found: 0.795 MRC and 0.821 LRC

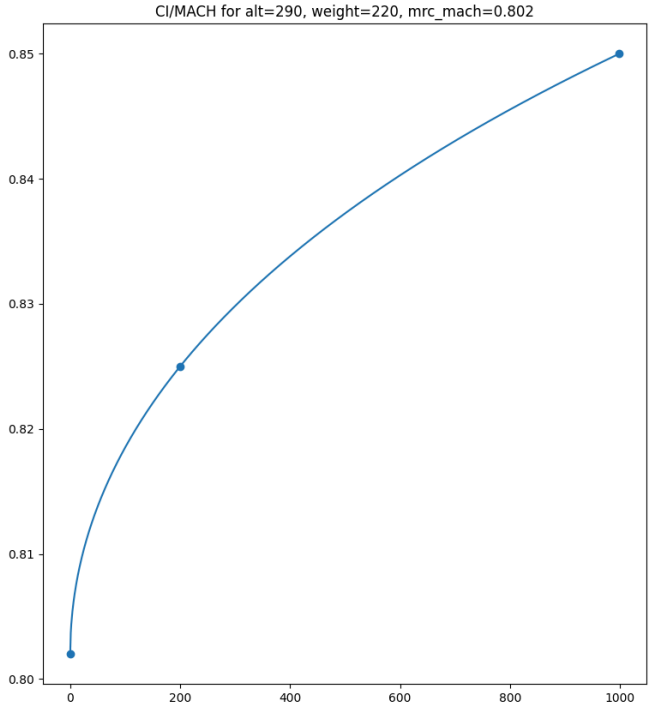




Sanity check: 0.795 MRC and 0.821 LRC

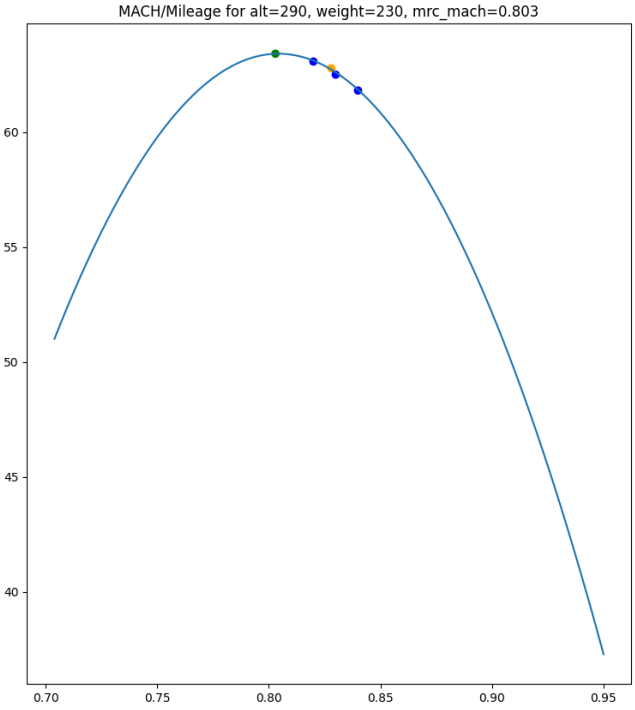
Found: 0.802 MRC and 0.825 LRC

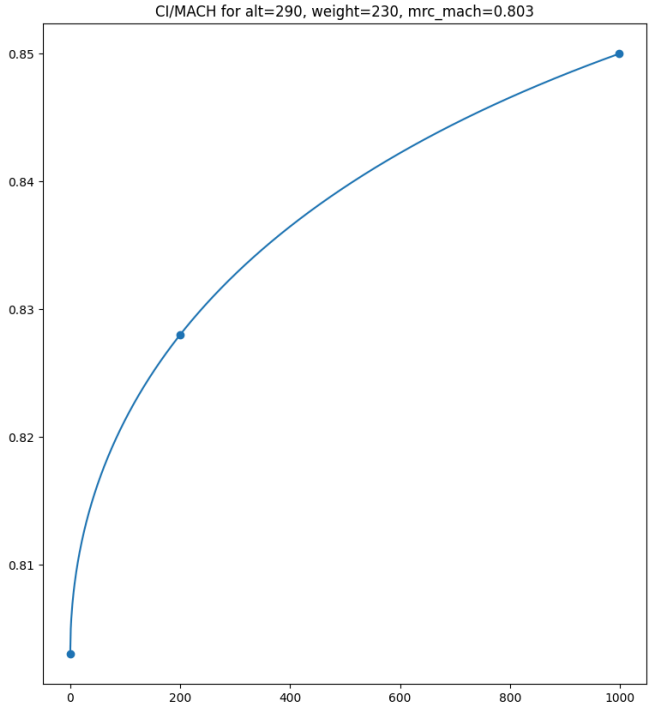




Sanity check: 0.802 MRC and 0.825 LRC

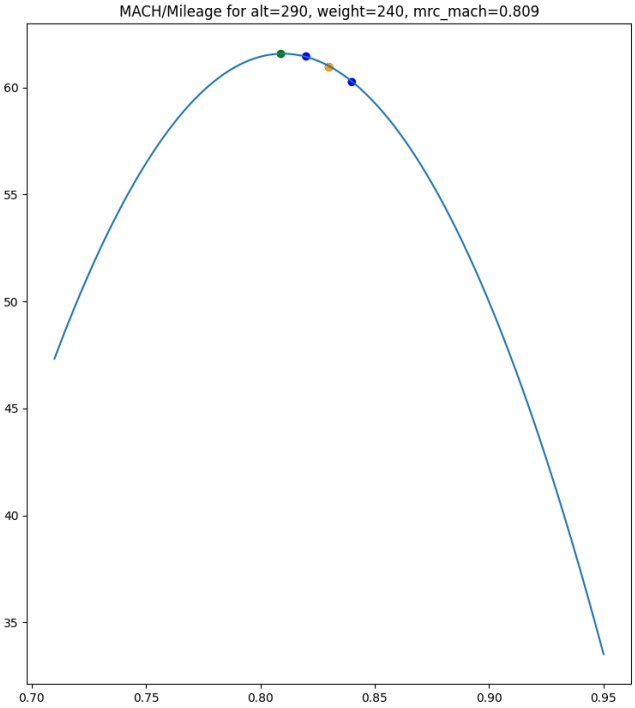
Found: 0.803 MRC and 0.828 LRC

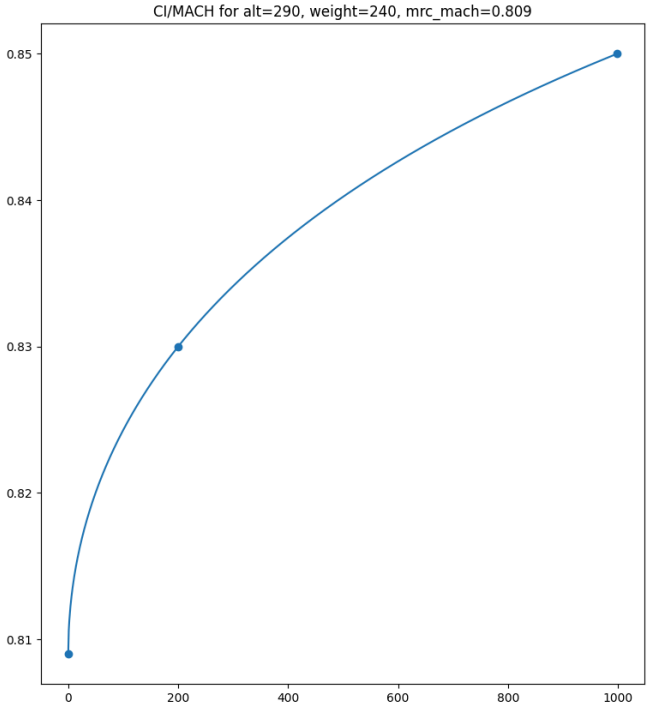




Sanity check: 0.803 MRC and 0.828 LRC

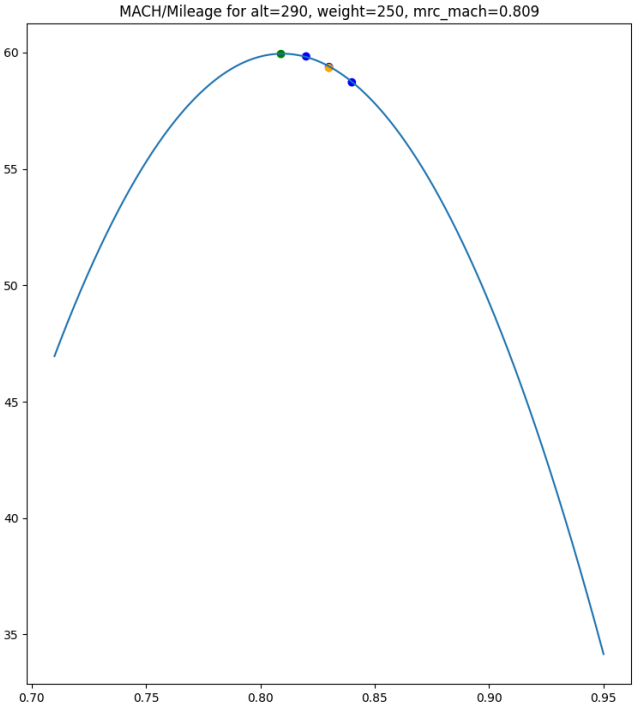
Found: 0.809 MRC and 0.83 LRC

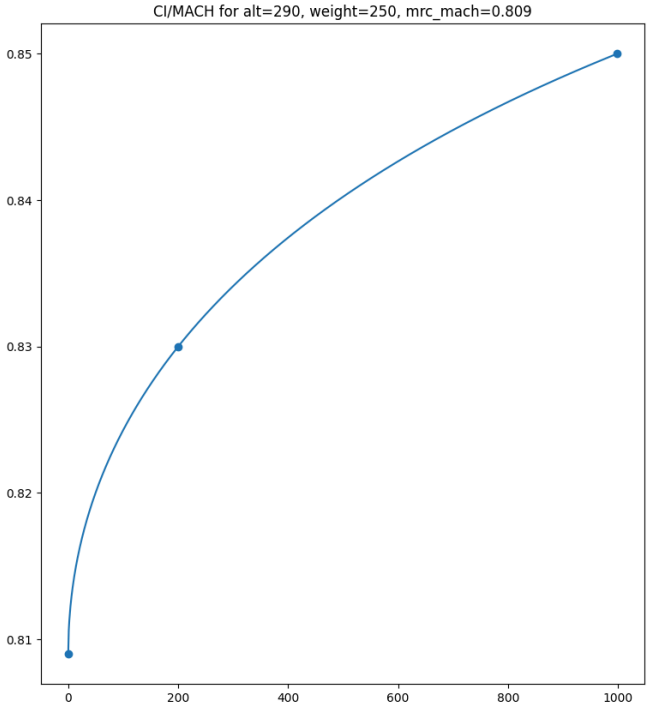




Sanity check: 0.809 MRC and 0.83 LRC

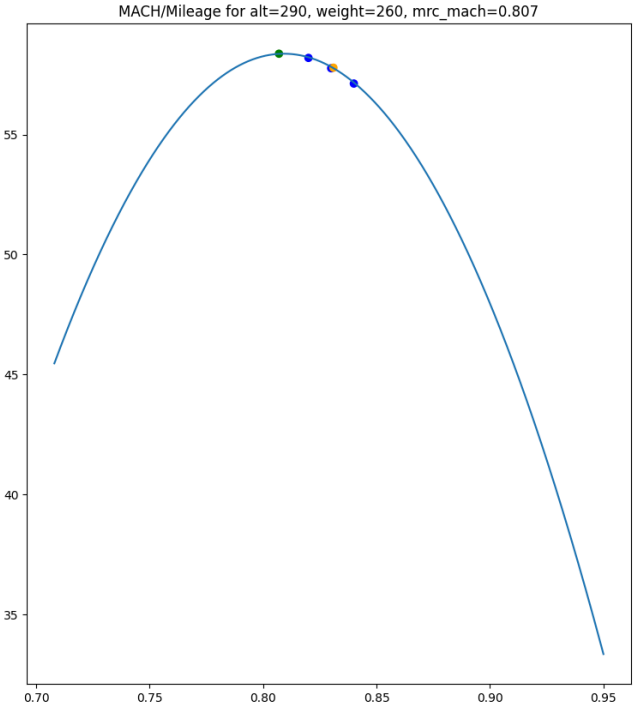
Found: 0.809 MRC and 0.83 LRC

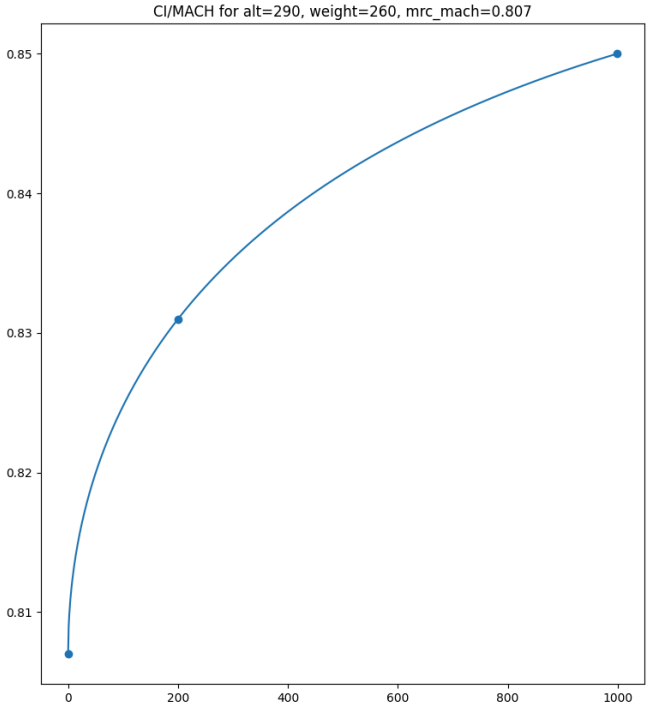




Sanity check: 0.809 MRC and 0.83 LRC

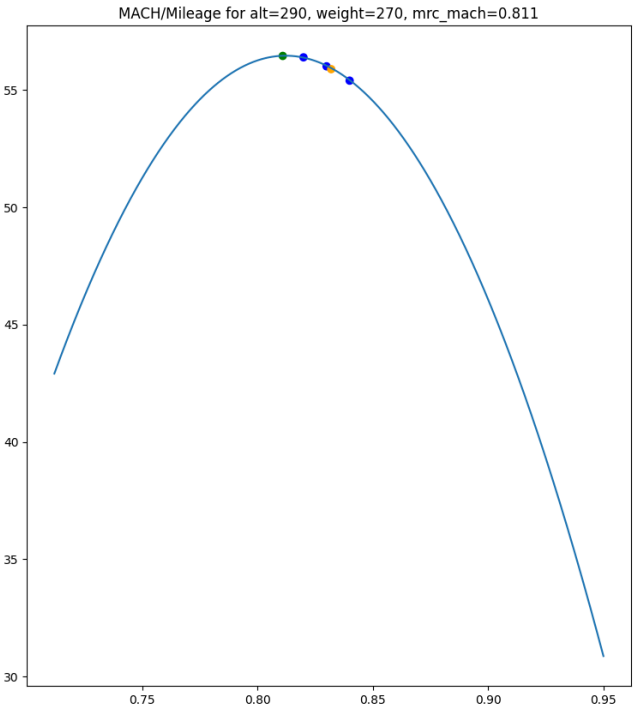
Found: 0.807 MRC and 0.831 LRC

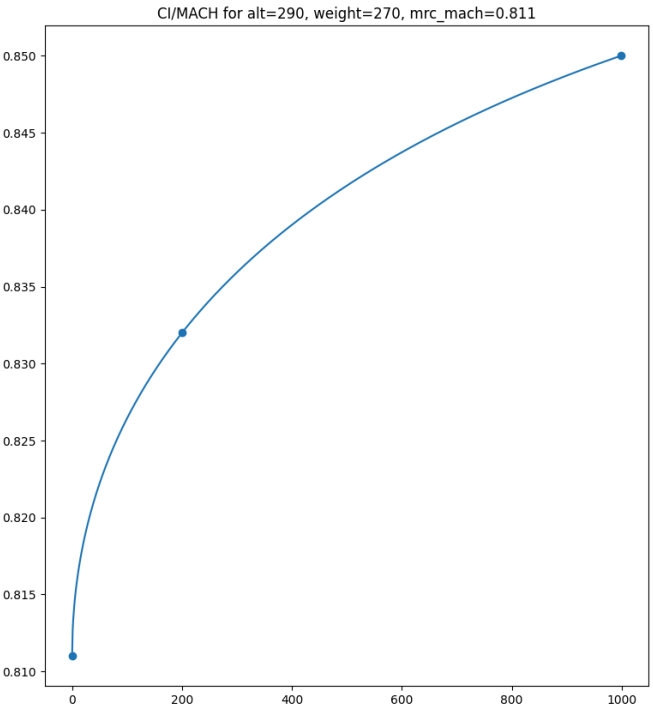




Sanity check: 0.807 MRC and 0.831 LRC

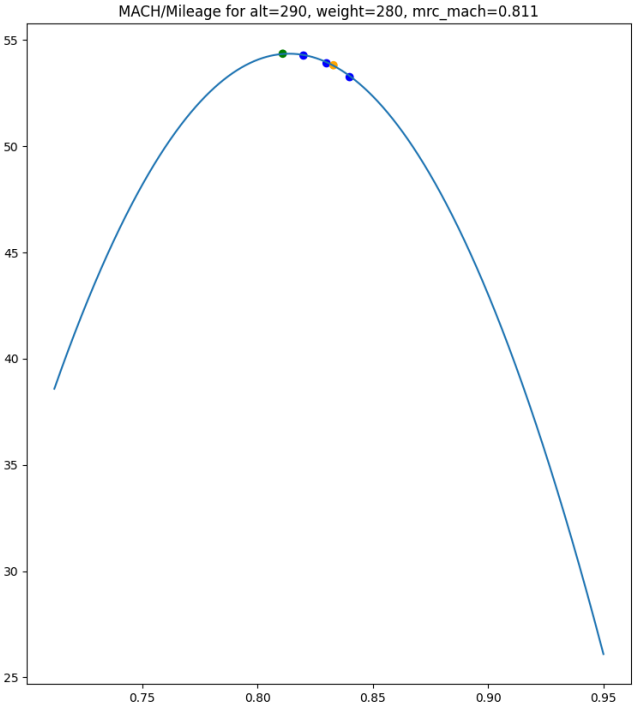
Found: 0.811 MRC and 0.832 LRC

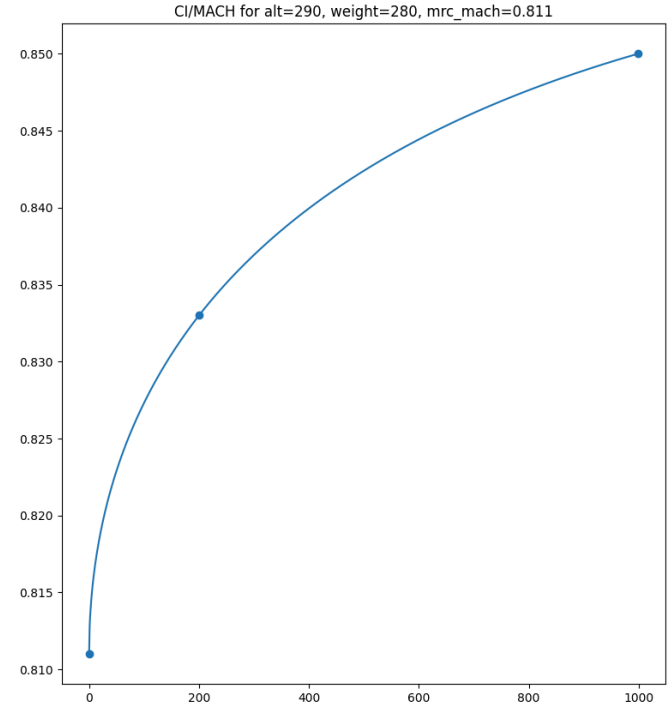




Sanity check: 0.811 MRC and 0.832 LRC

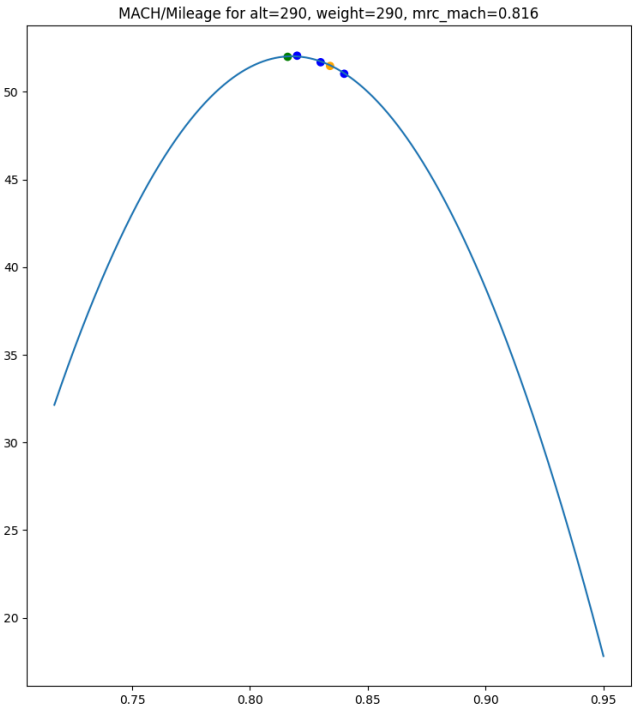
Found: 0.811 MRC and 0.833 LRC

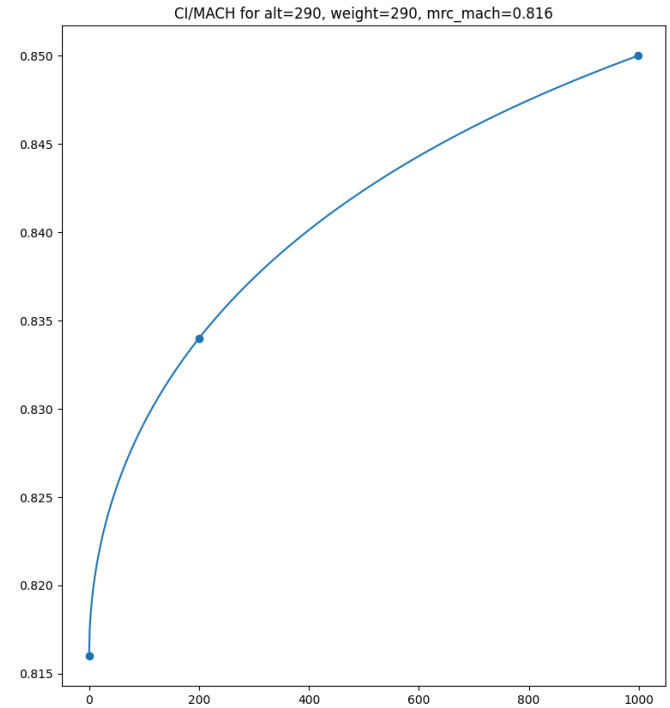




Sanity check: 0.811 MRC and 0.833 LRC

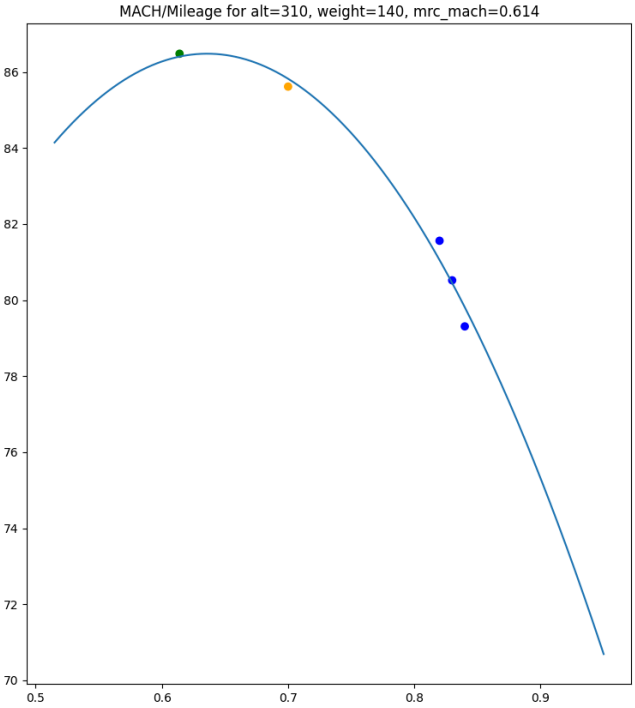
Found: 0.816 MRC and 0.834 LRC

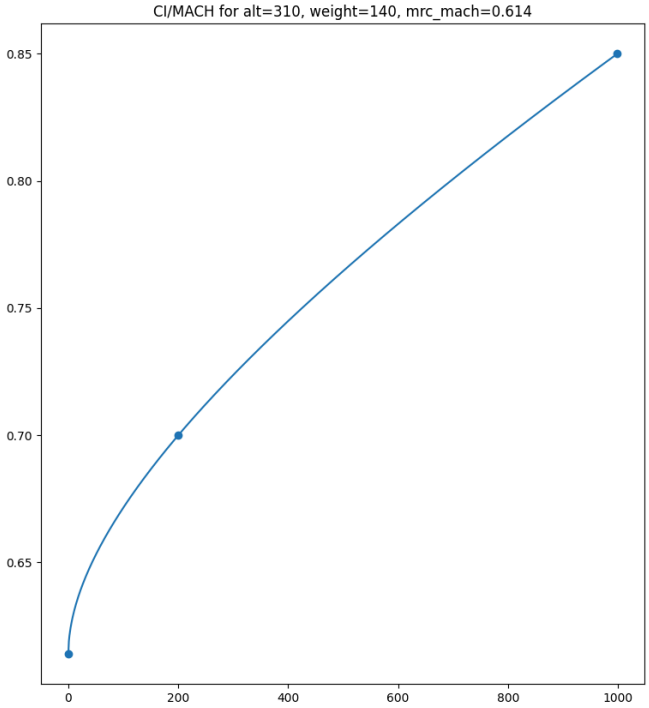




Sanity check: 0.816 MRC and 0.834 LRC

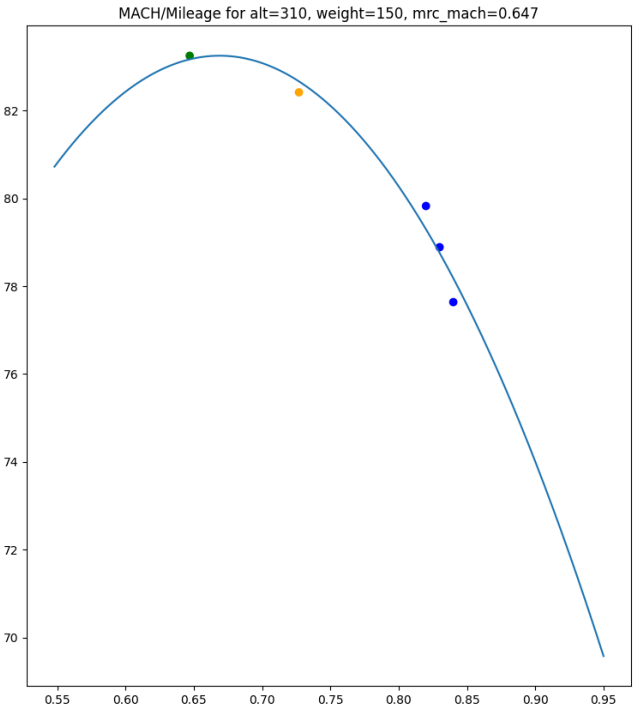
Found: 0.614 MRC and 0.7 LRC

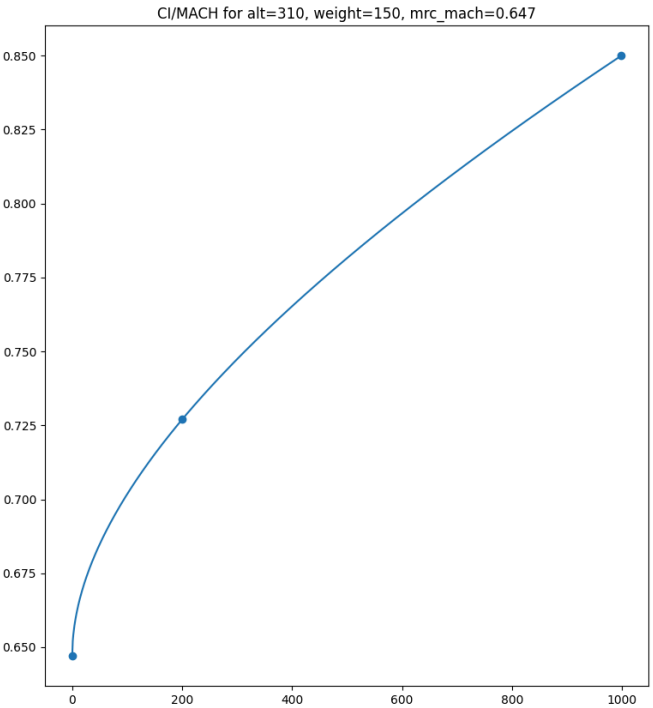




Sanity check: 0.614 MRC and 0.7 LRC

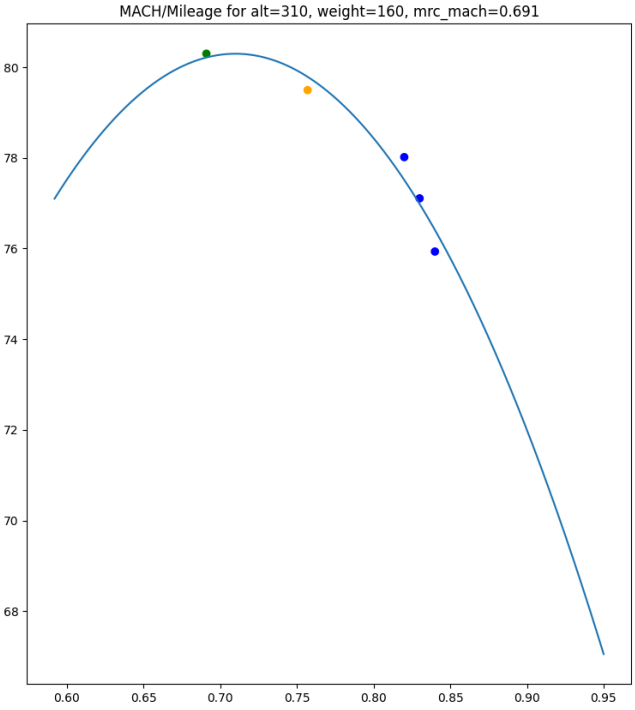
Found: 0.647 MRC and 0.727 LRC

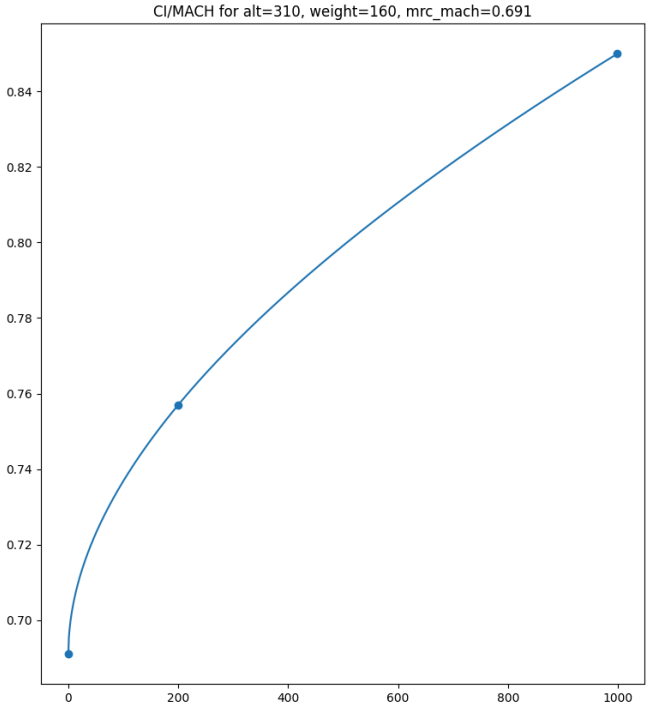




Sanity check: 0.647 MRC and 0.727 LRC

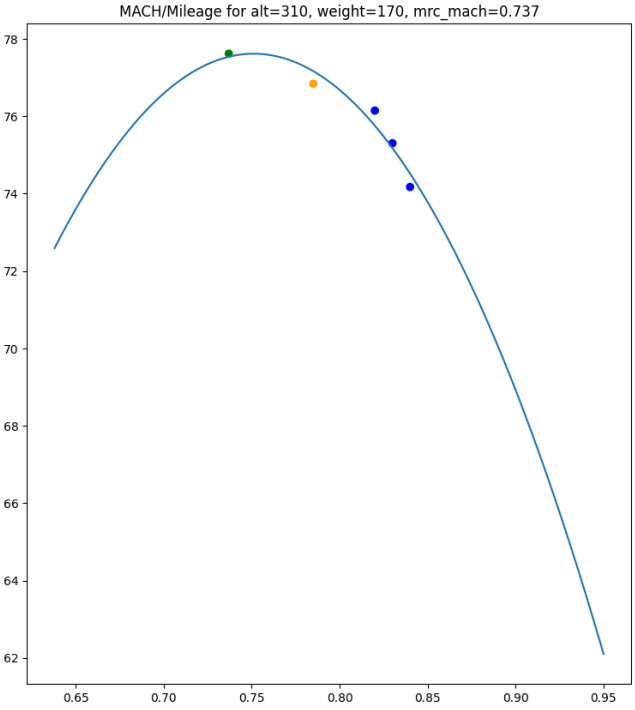
Found: 0.691 MRC and 0.757 LRC

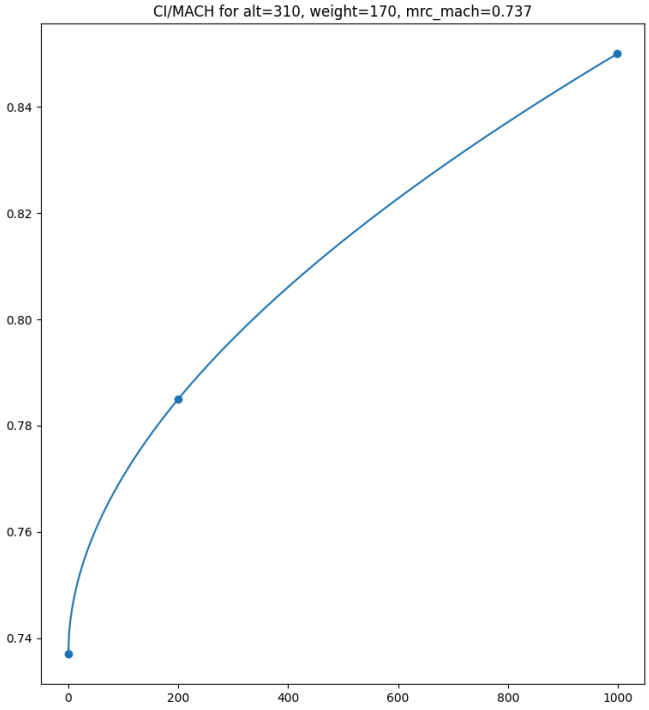




Sanity check: 0.691 MRC and 0.757 LRC

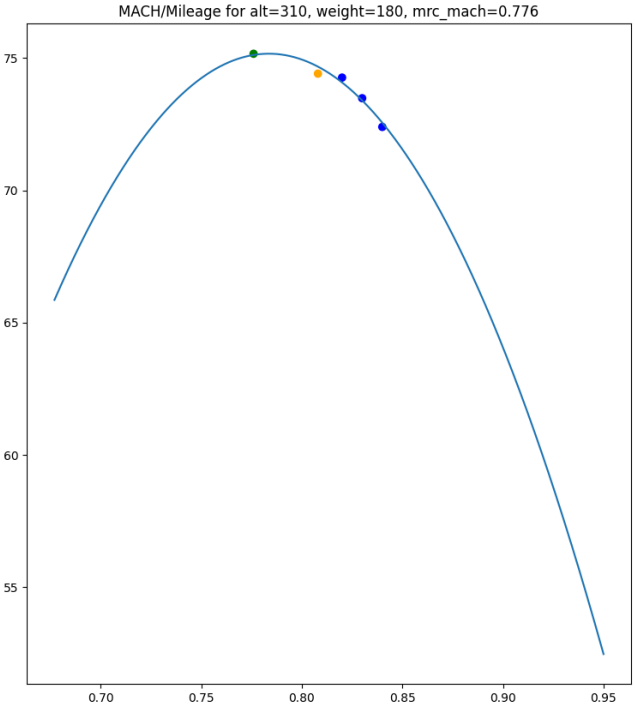
Found: 0.737 MRC and 0.785 LRC

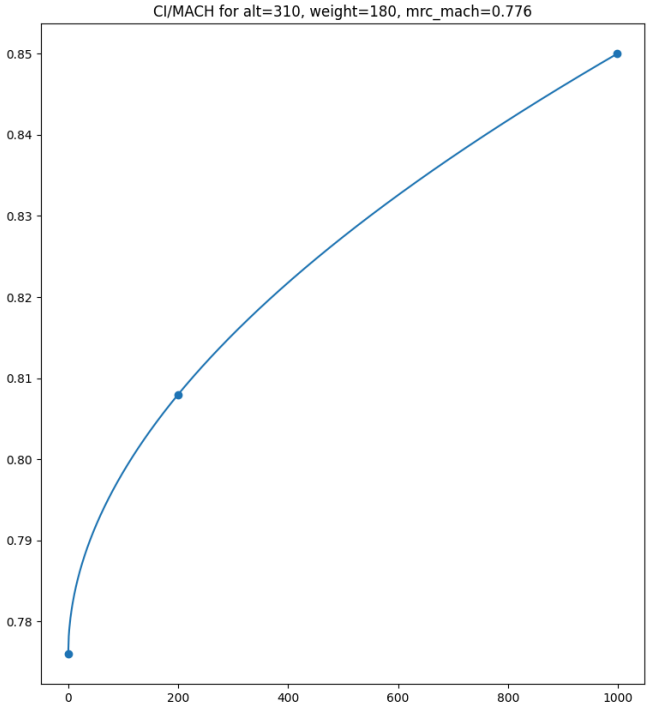




Sanity check: 0.737 MRC and 0.785 LRC

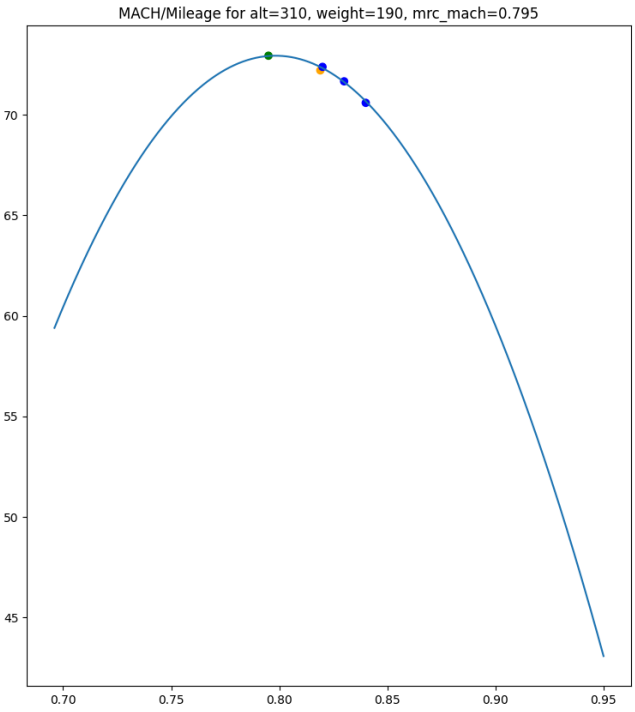
Found: 0.776 MRC and 0.808 LRC

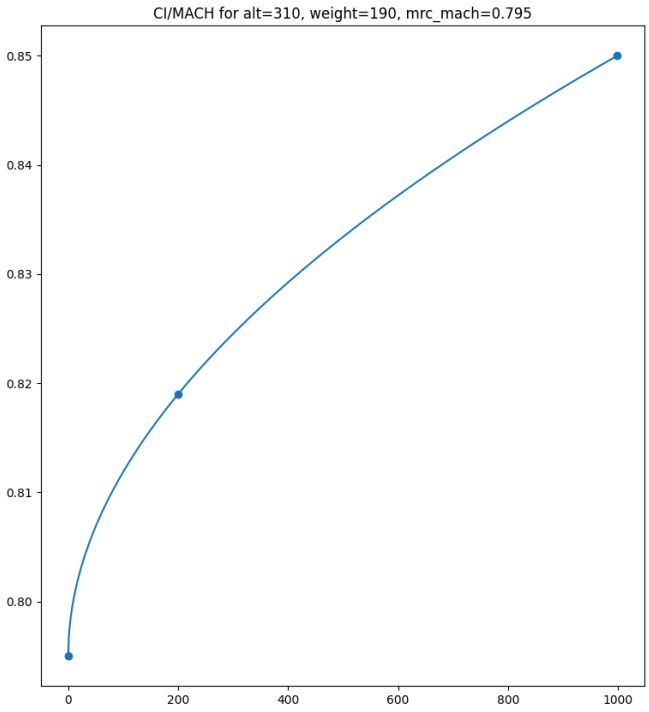




Sanity check: 0.776 MRC and 0.808 LRC

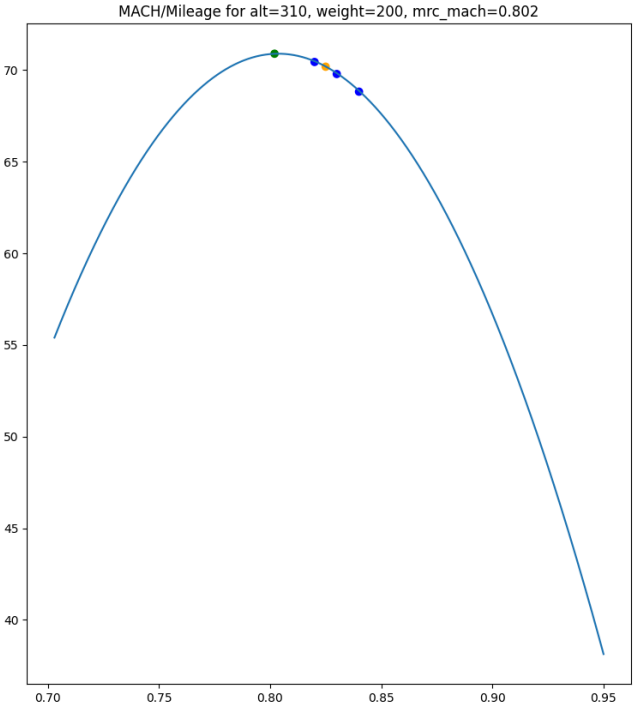
Found: 0.795 MRC and 0.819 LRC

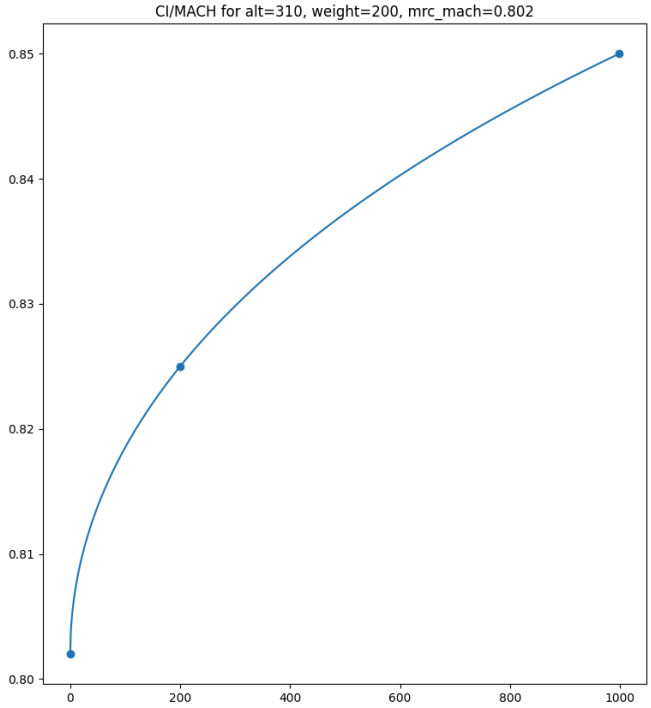




Sanity check: 0.795 MRC and 0.819 LRC

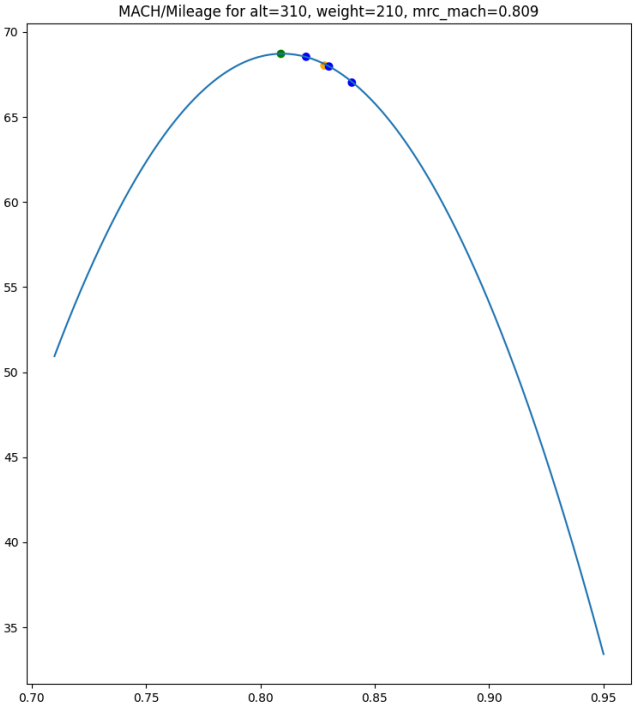
Found: 0.802 MRC and 0.825 LRC

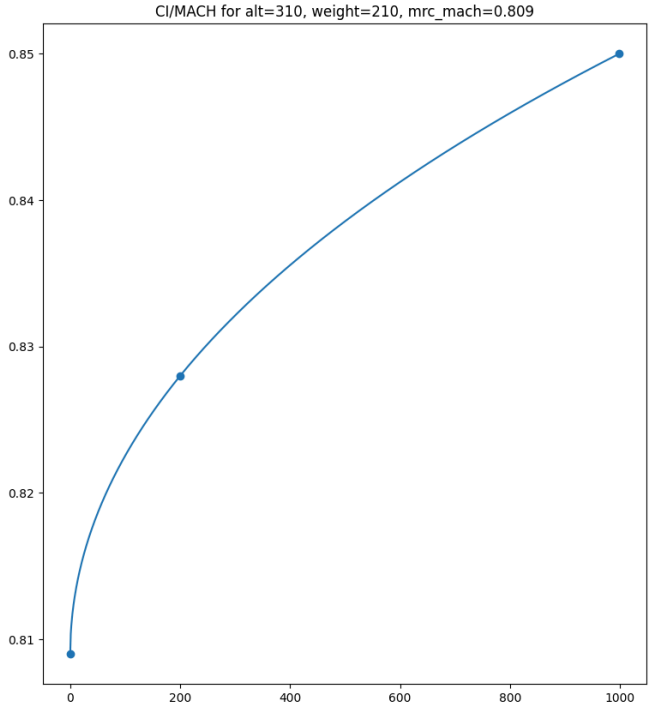




Sanity check: 0.802 MRC and 0.825 LRC

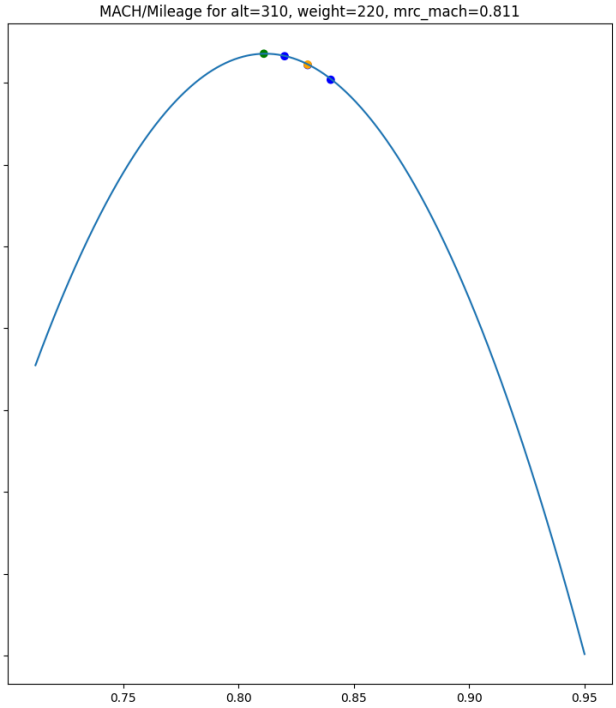
Found: 0.809 MRC and 0.828 LRC

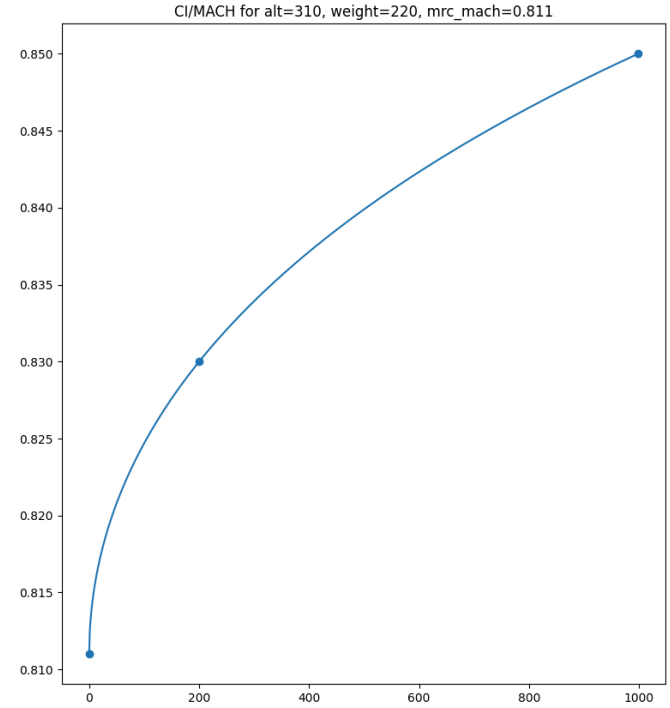




Sanity check: 0.809 MRC and 0.828 LRC

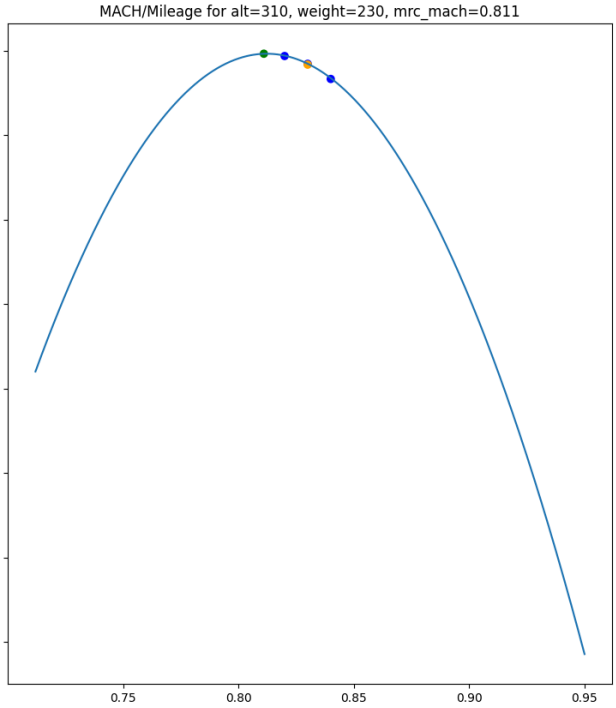
Found: 0.811 MRC and 0.83 LRC

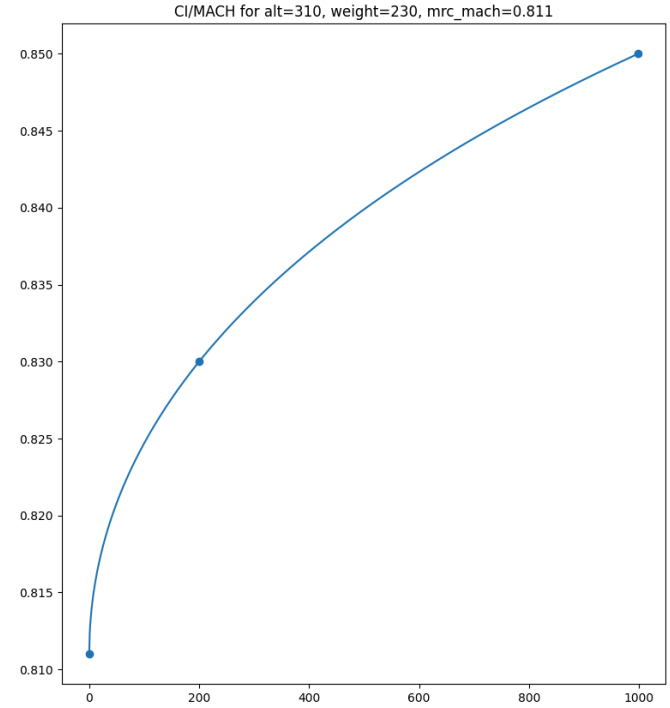




Sanity check: 0.811 MRC and 0.83 LRC

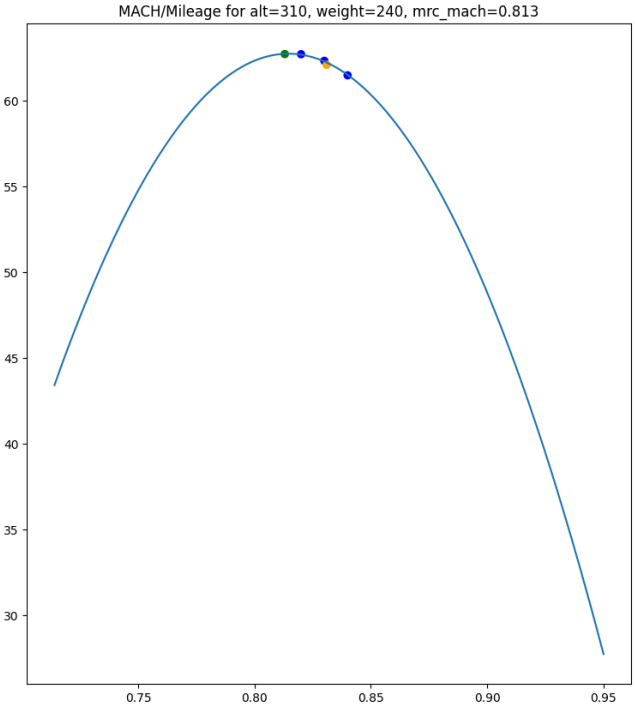
Found: 0.811 MRC and 0.83 LRC

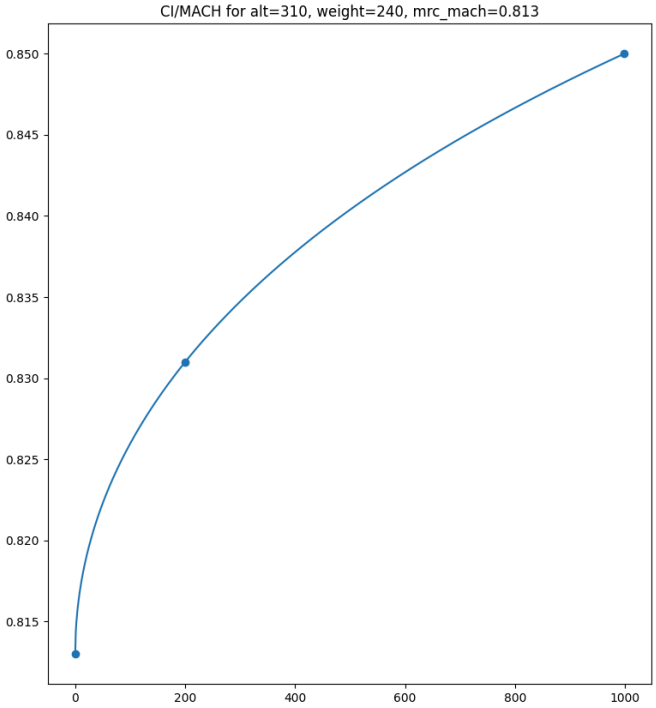




Sanity check: 0.811 MRC and 0.83 LRC

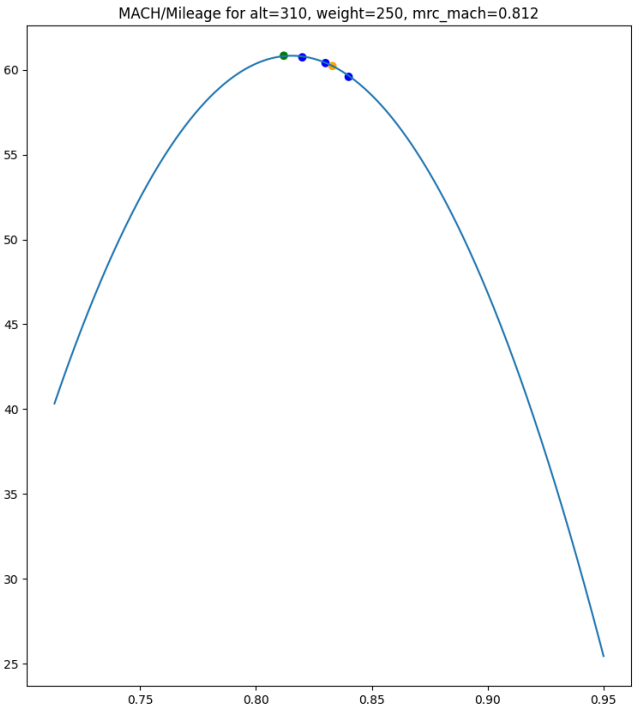
Found: 0.813 MRC and 0.831 LRC

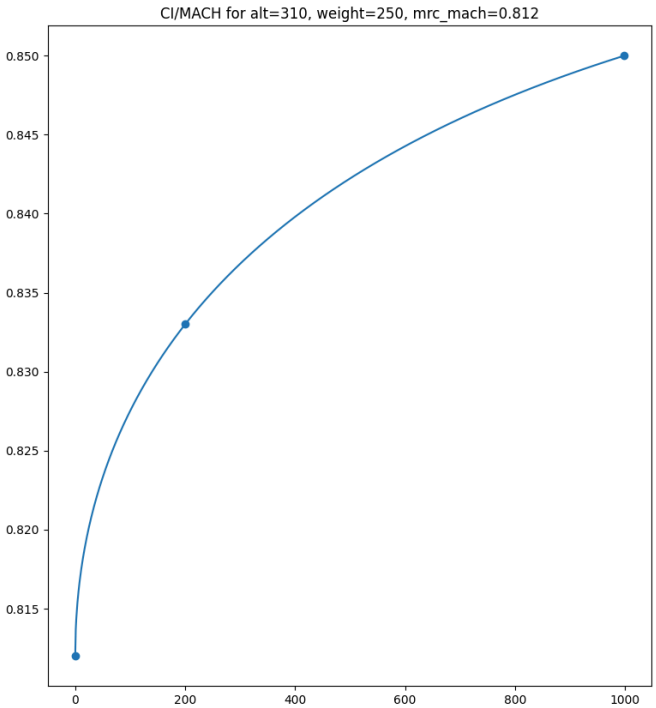




Sanity check: 0.813 MRC and 0.831 LRC

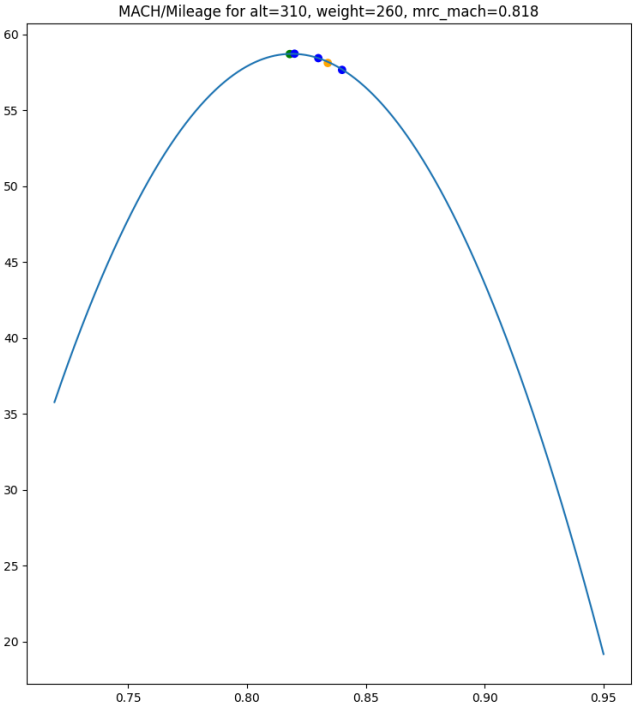
Found: 0.812 MRC and 0.833 LRC

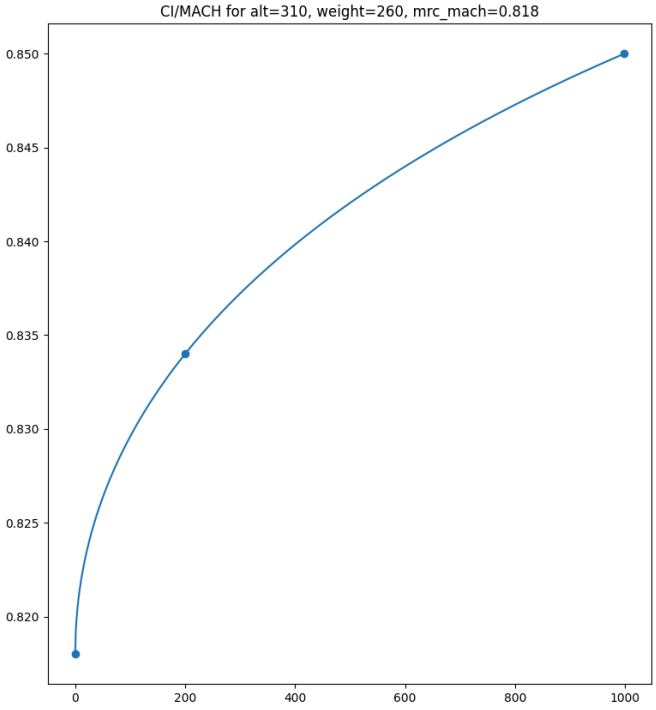




Sanity check: 0.812 MRC and 0.833 LRC

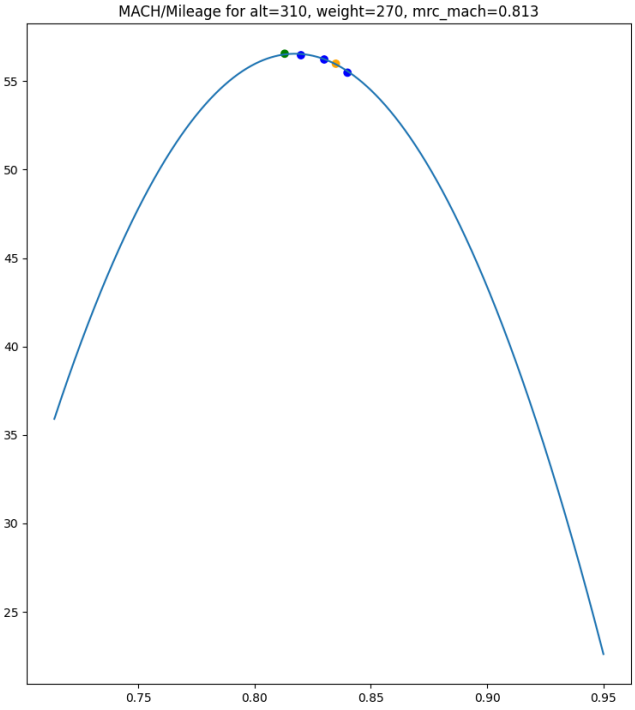
Found: 0.818 MRC and 0.834 LRC

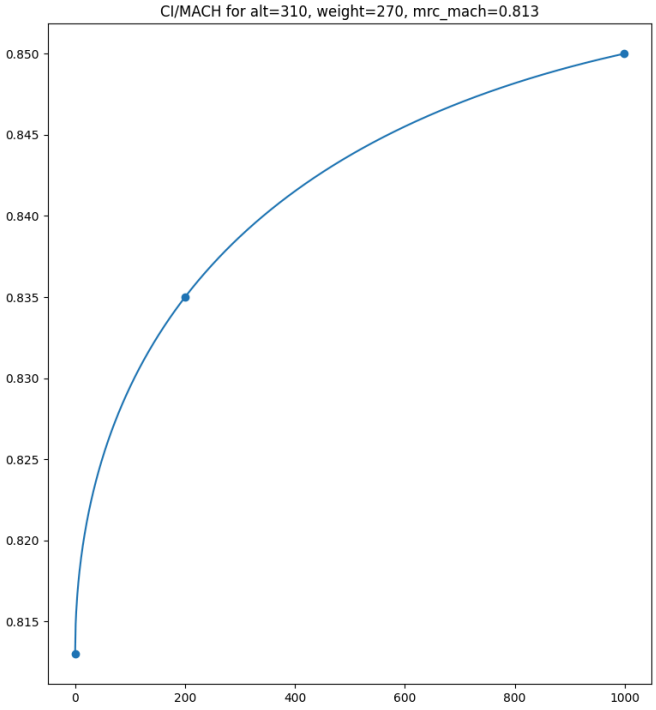




Sanity check: 0.818 MRC and 0.834 LRC

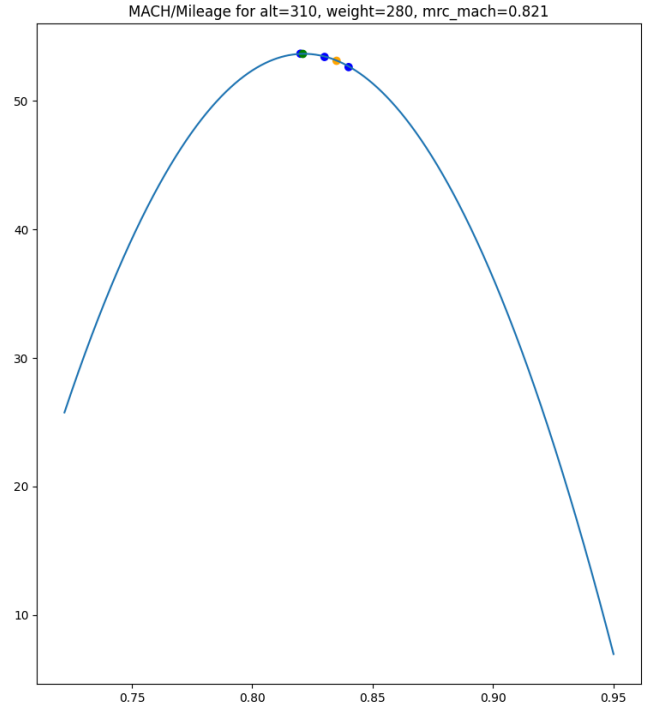
Found: 0.813 MRC and 0.835 LRC

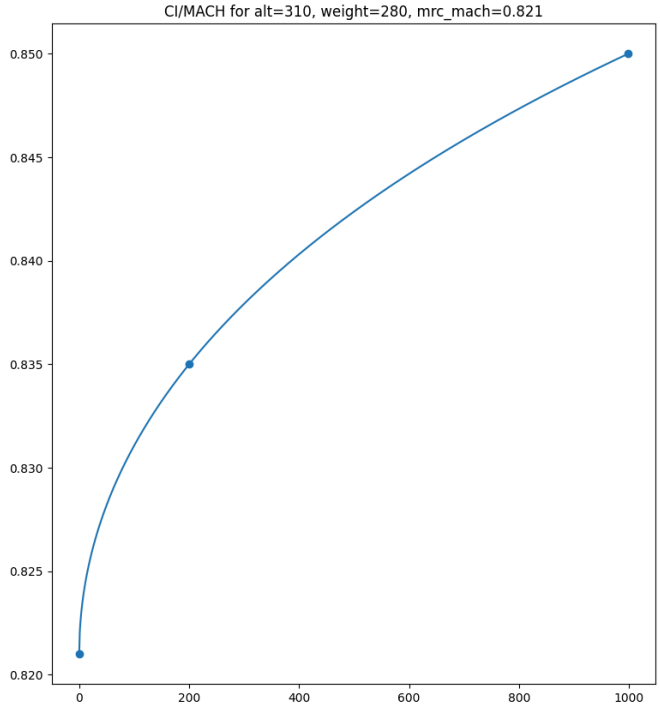




Sanity check: 0.813 MRC and 0.835 LRC

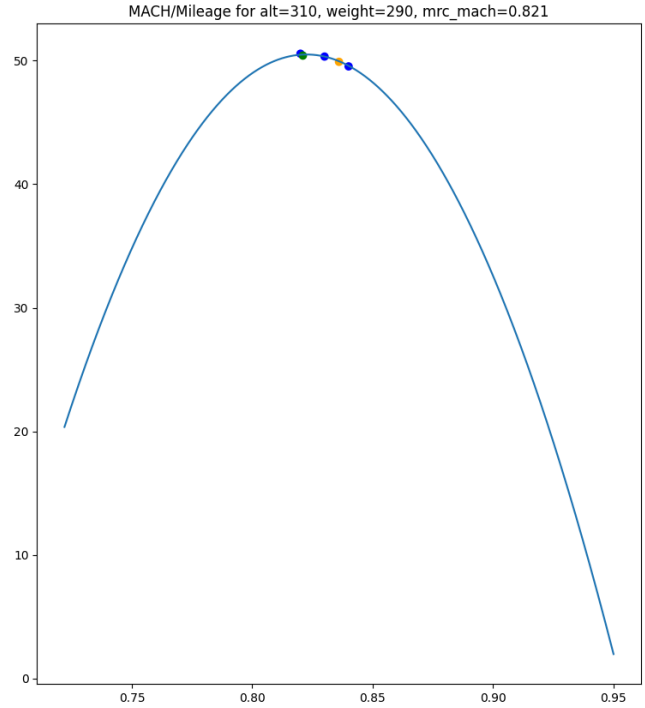
Found: 0.821 MRC and 0.835 LRC

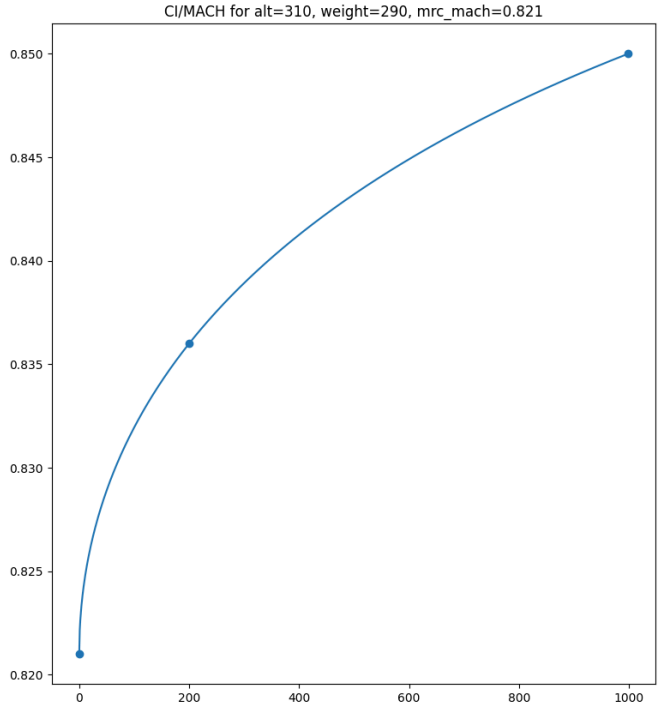




Sanity check: 0.821 MRC and 0.835 LRC

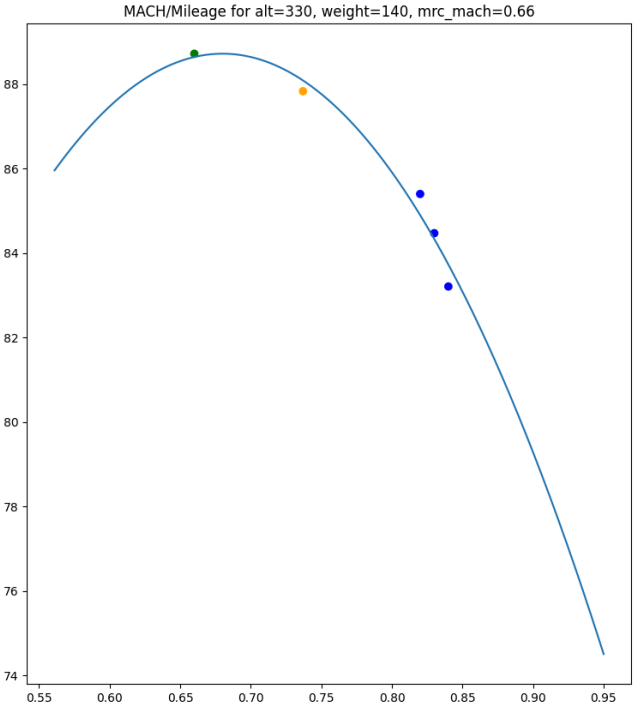
Found: 0.821 MRC and 0.836 LRC

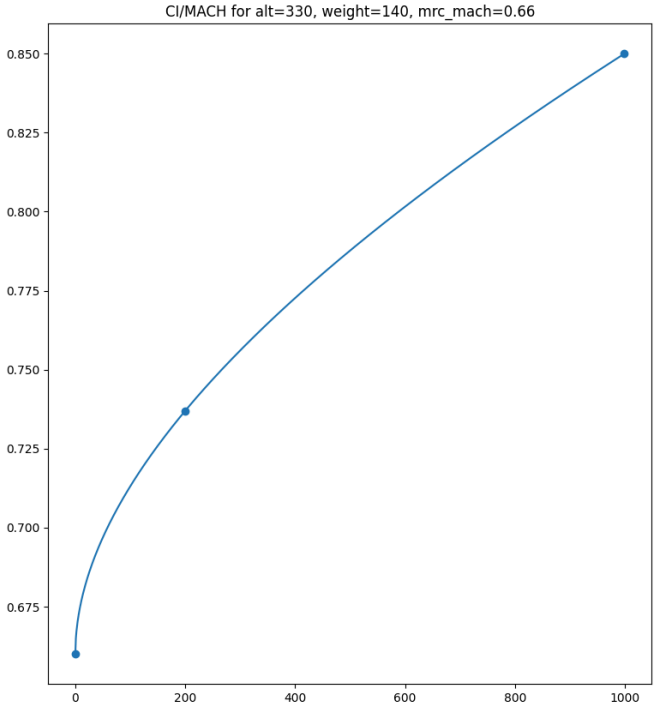




Sanity check: 0.821 MRC and 0.836 LRC

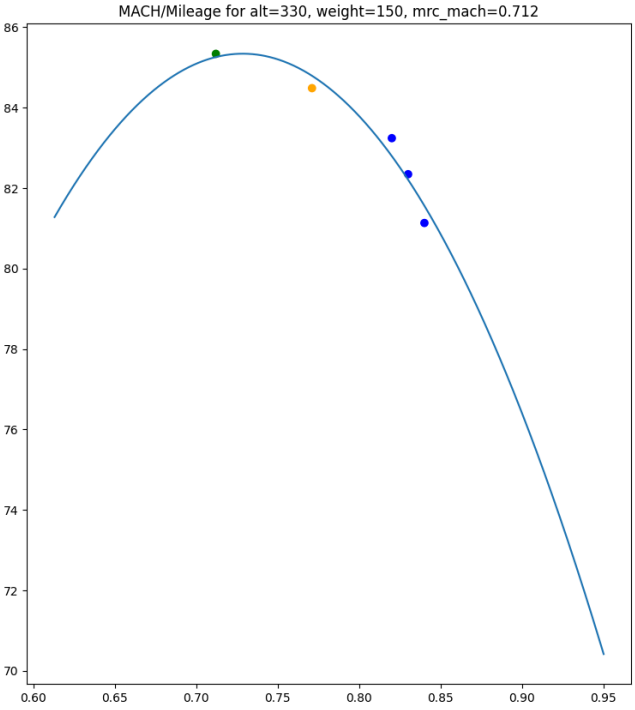
Found: 0.66 MRC and 0.737 LRC

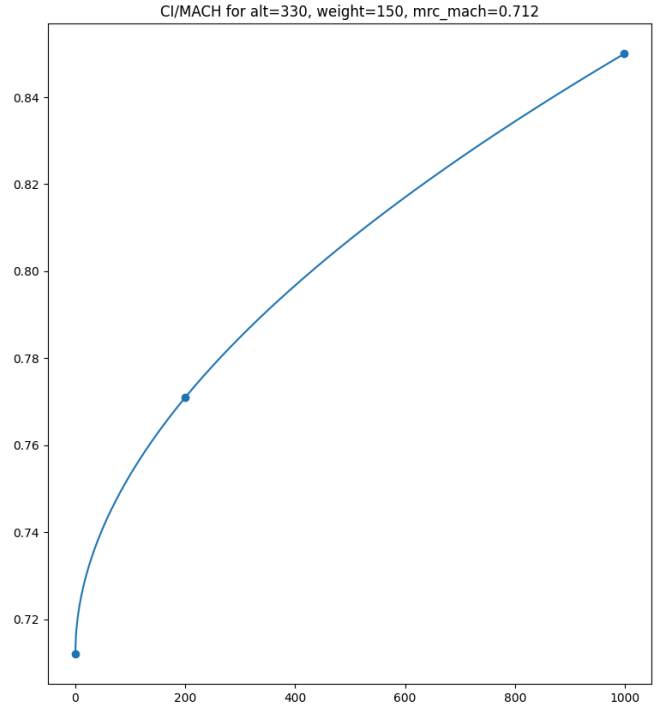




Sanity check: 0.66 MRC and 0.737 LRC

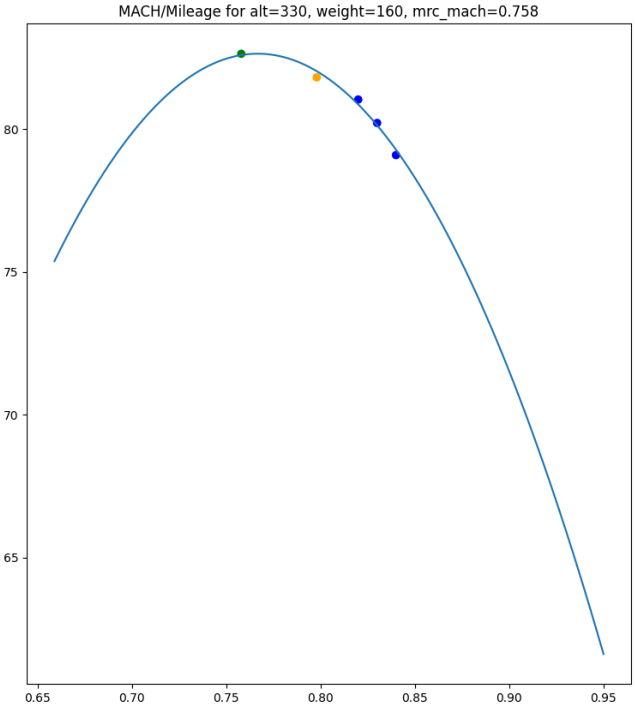
Found: 0.712 MRC and 0.771 LRC

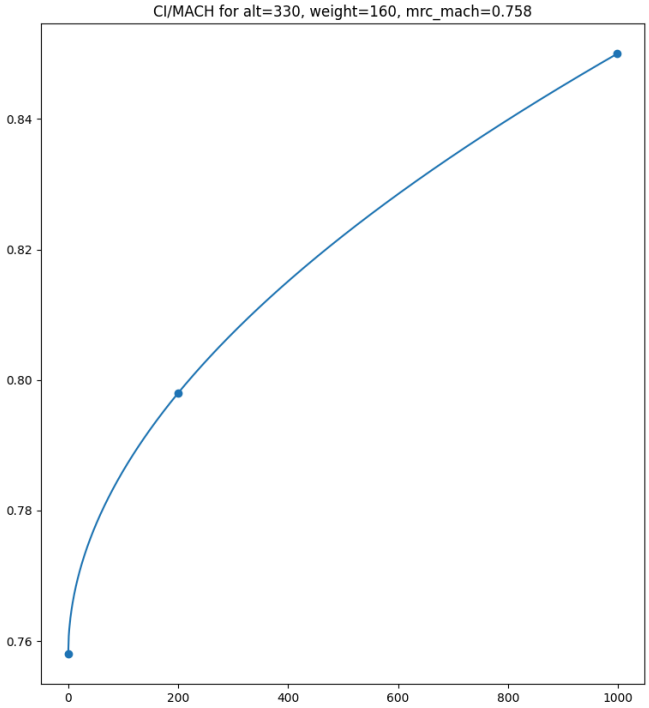




Sanity check: 0.712 MRC and 0.771 LRC

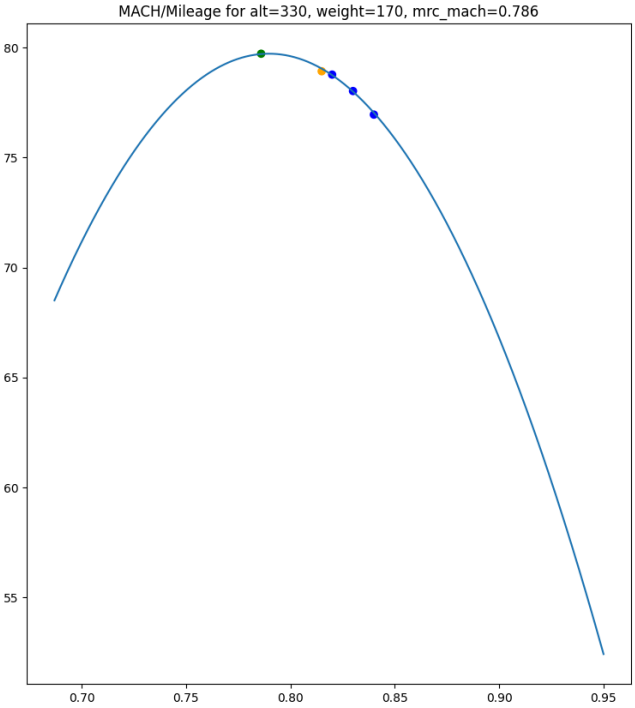
Found: 0.758 MRC and 0.798 LRC

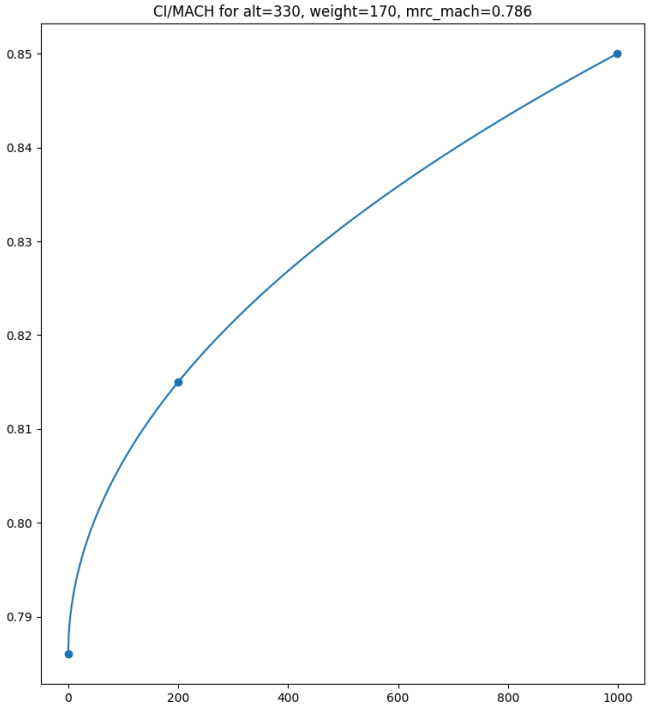




Sanity check: 0.758 MRC and 0.798 LRC

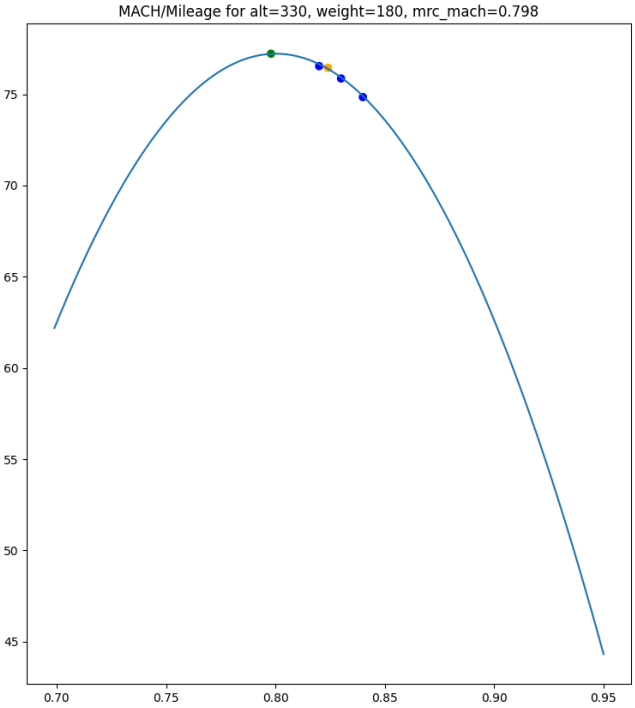
Found: 0.786 MRC and 0.815 LRC

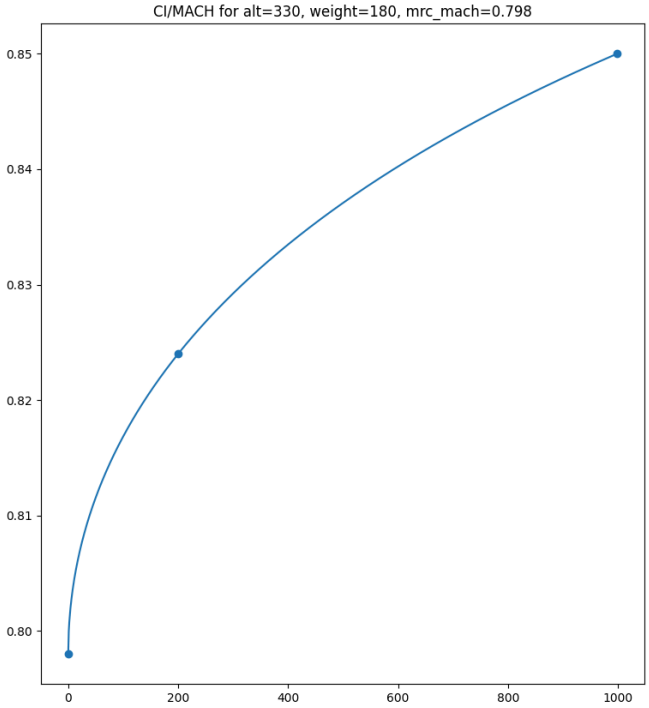




Sanity check: 0.786 MRC and 0.815 LRC

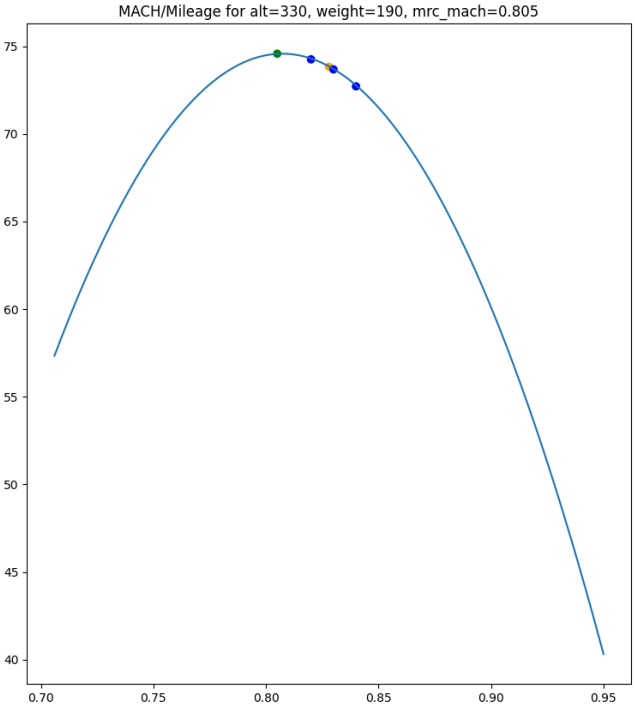
Found: 0.798 MRC and 0.824 LRC

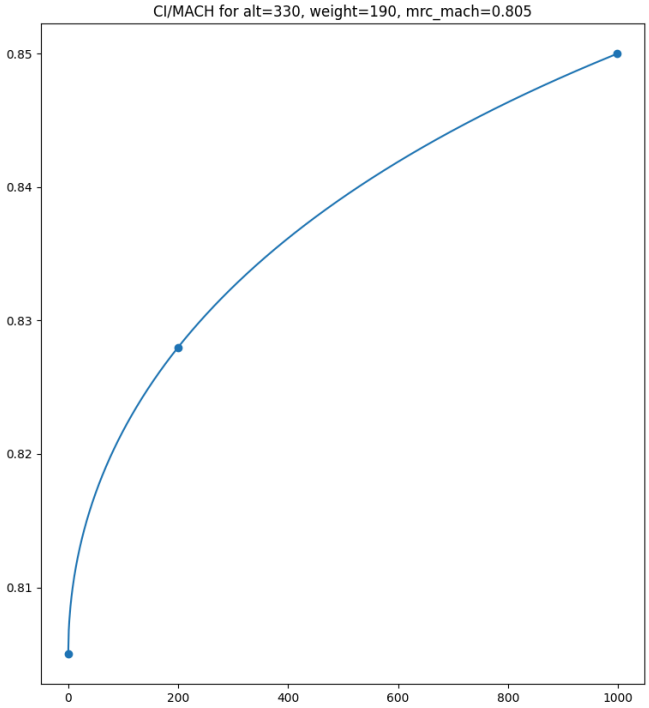




Sanity check: 0.798 MRC and 0.824 LRC

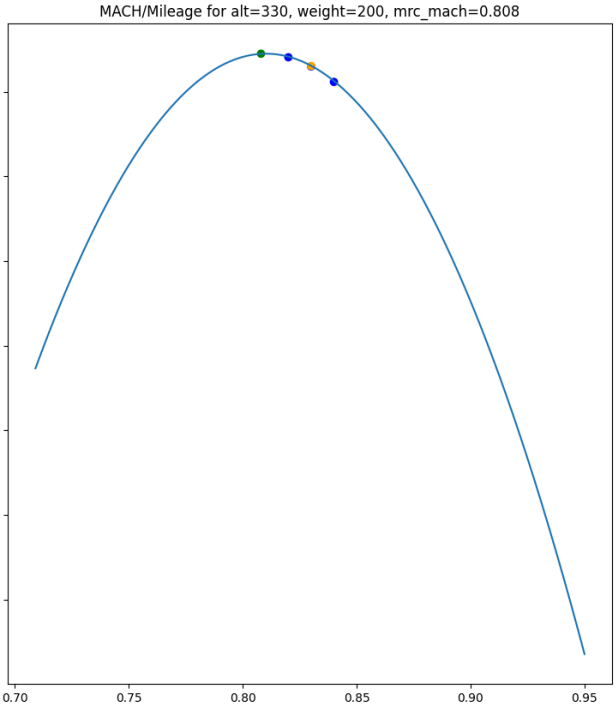
Found: 0.805 MRC and 0.828 LRC

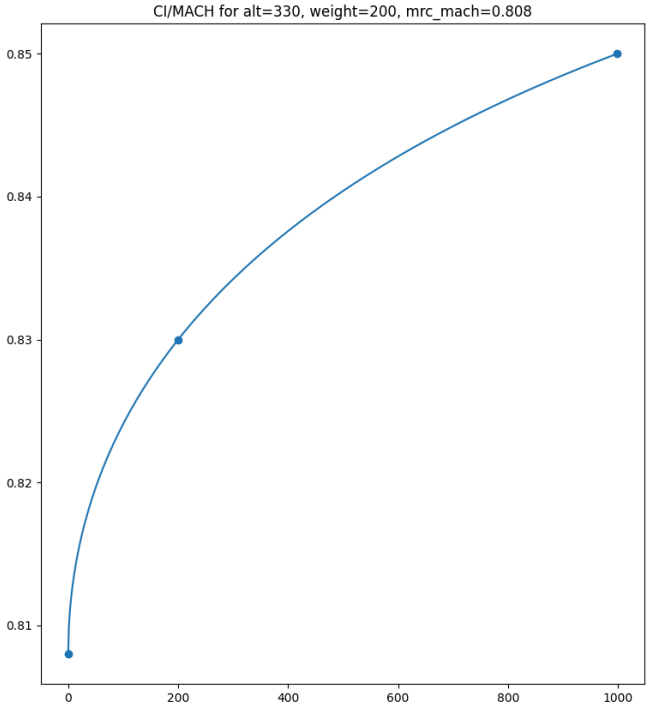




Sanity check: 0.805 MRC and 0.828 LRC

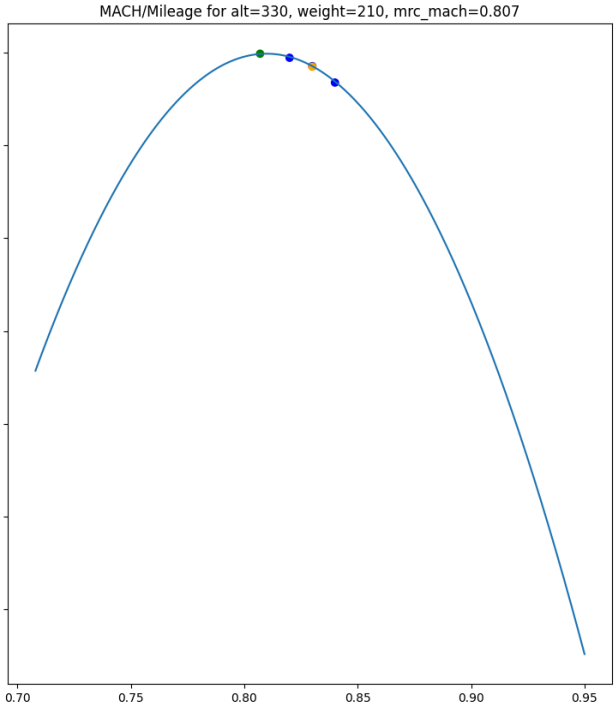
Found: 0.808 MRC and 0.83 LRC

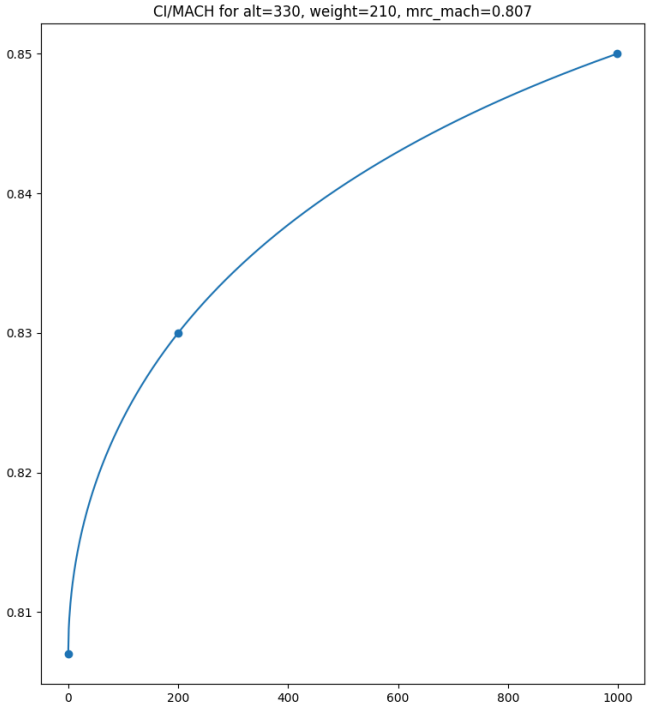




Sanity check: 0.808 MRC and 0.83 LRC

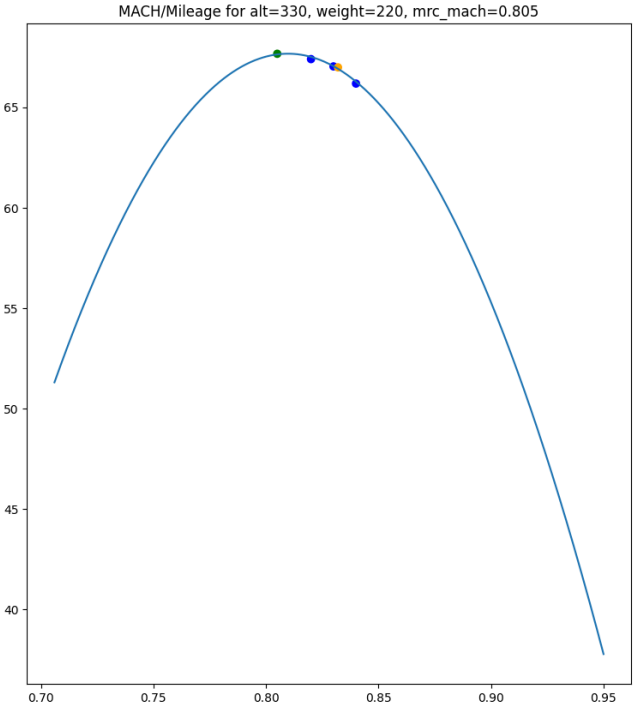
Found: 0.807 MRC and 0.83 LRC

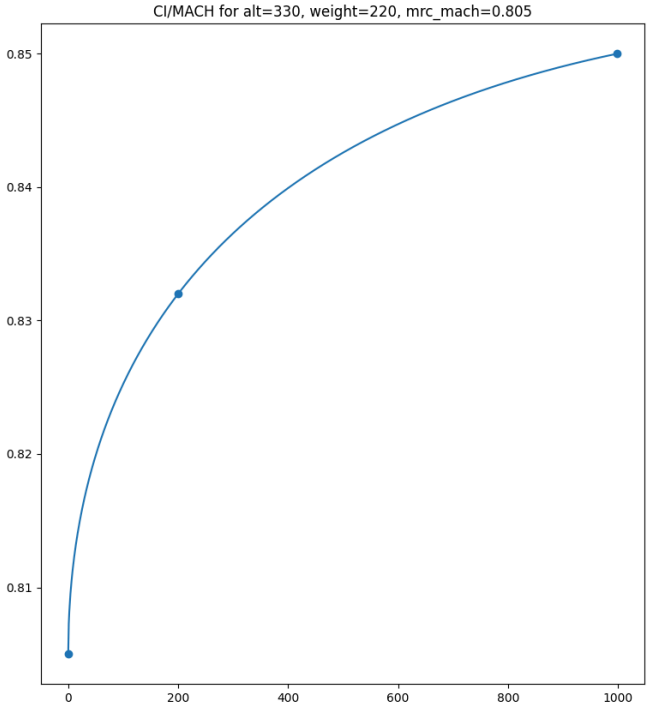




Sanity check: 0.807 MRC and 0.83 LRC

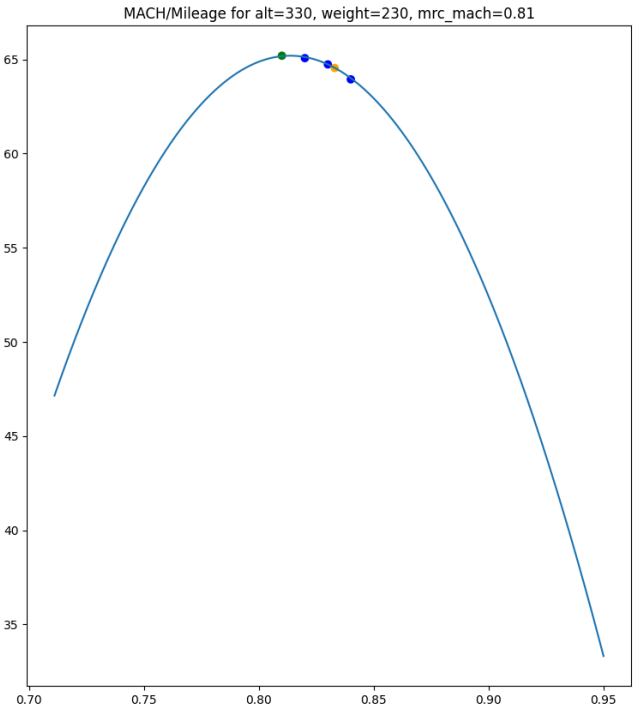
Found: 0.805 MRC and 0.832 LRC

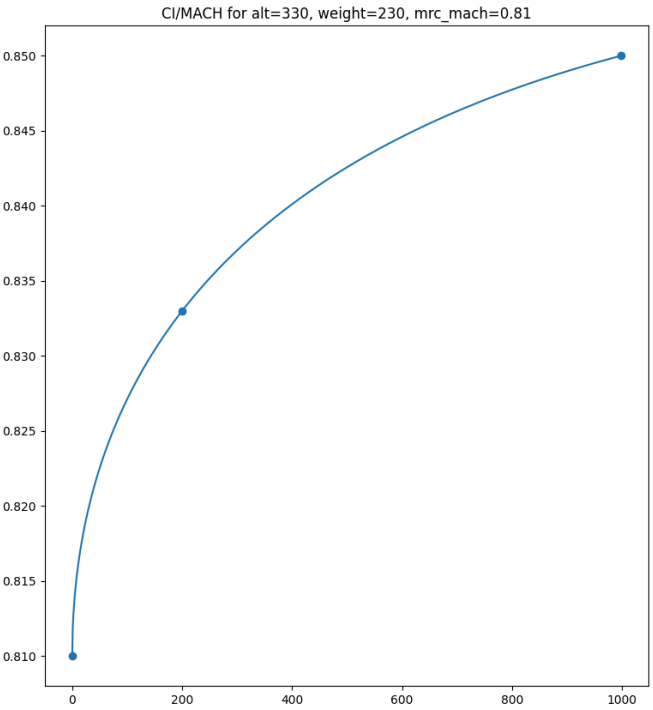




Sanity check: 0.805 MRC and 0.832 LRC

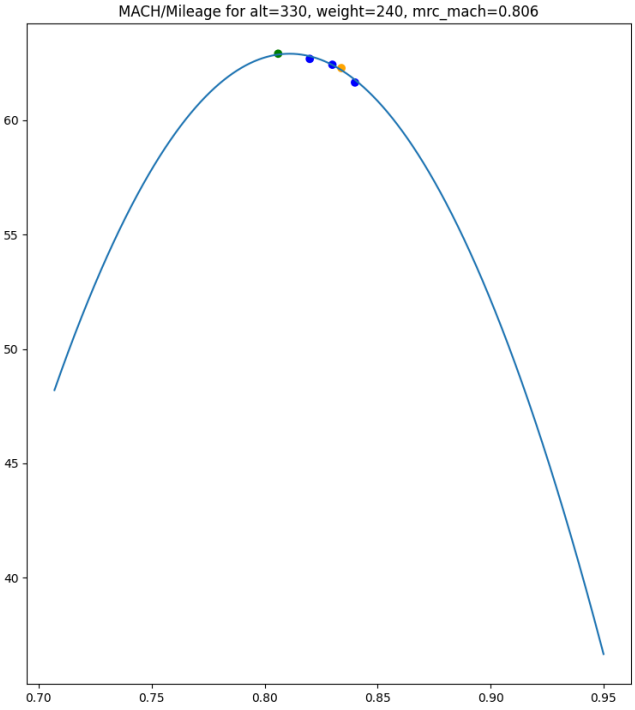
Found: 0.81 MRC and 0.833 LRC

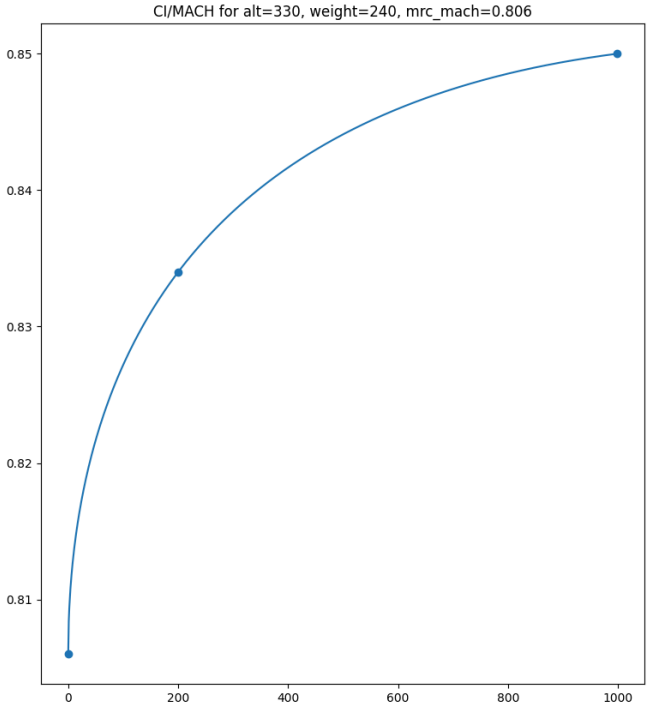




Sanity check: 0.81 MRC and 0.833 LRC

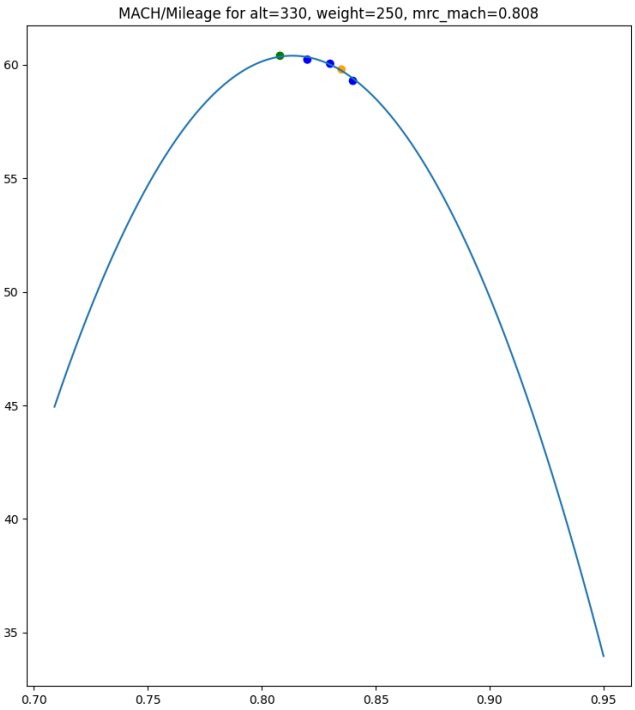
Found: 0.806 MRC and 0.834 LRC

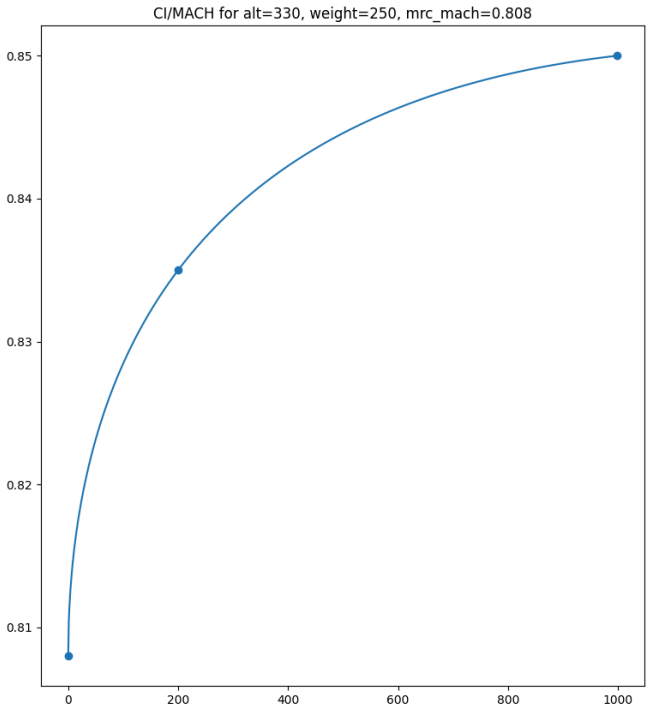




Sanity check: 0.806 MRC and 0.834 LRC

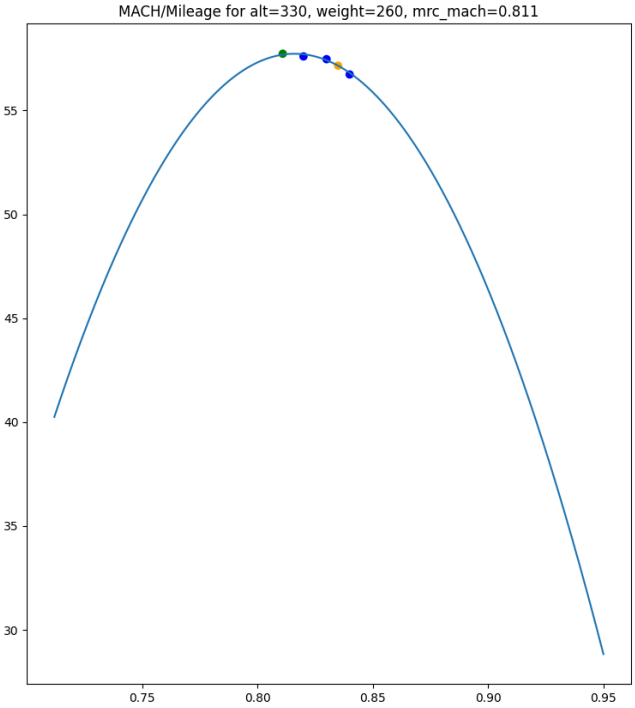
Found: 0.808 MRC and 0.835 LRC

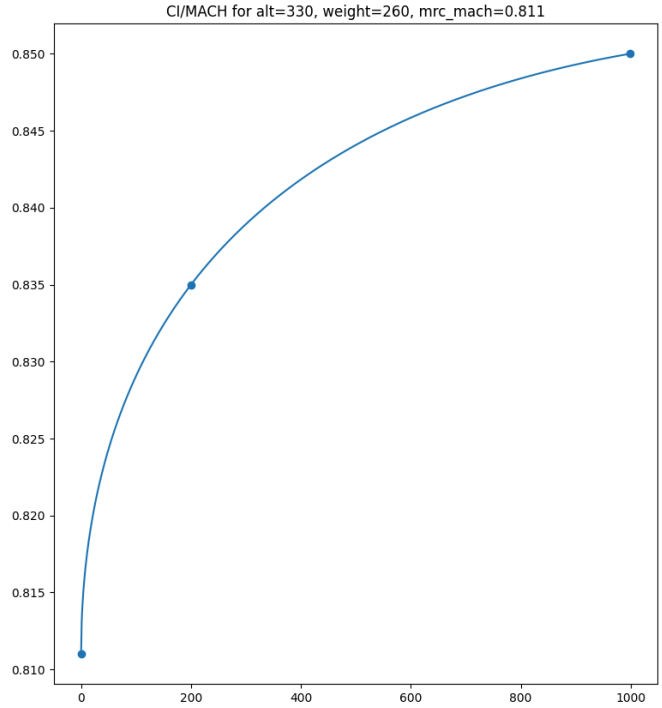




Sanity check: 0.808 MRC and 0.835 LRC

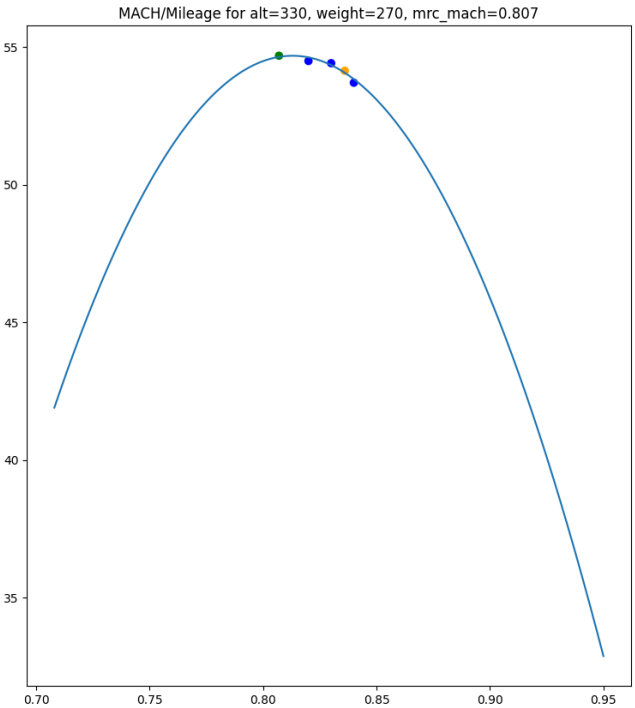
Found: 0.811 MRC and 0.835 LRC

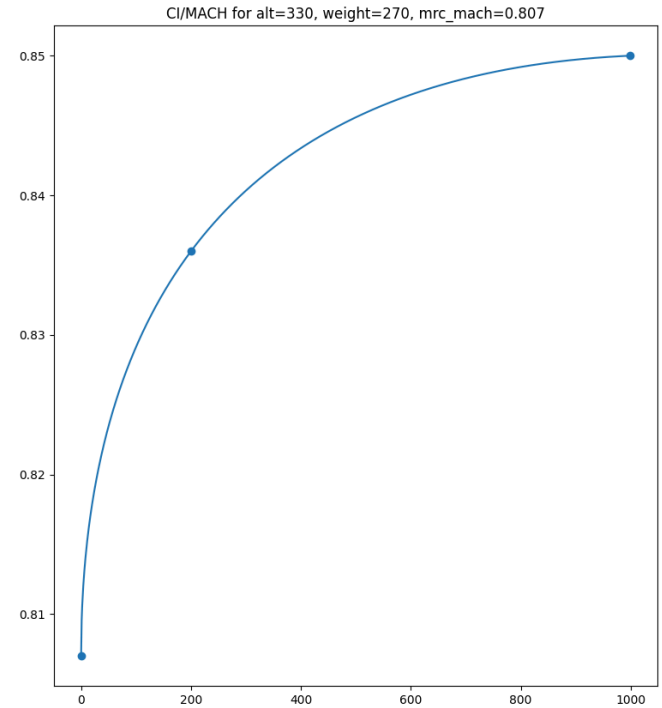




Sanity check: 0.811 MRC and 0.835 LRC

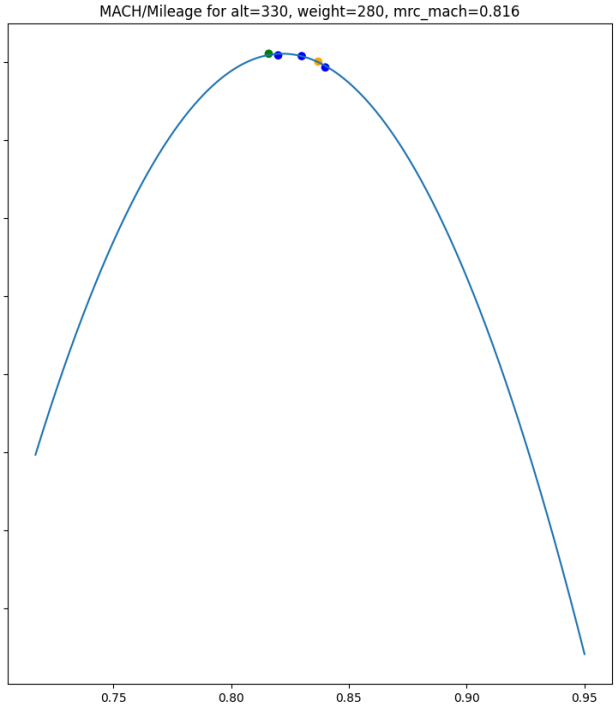
Found: 0.807 MRC and 0.836 LRC

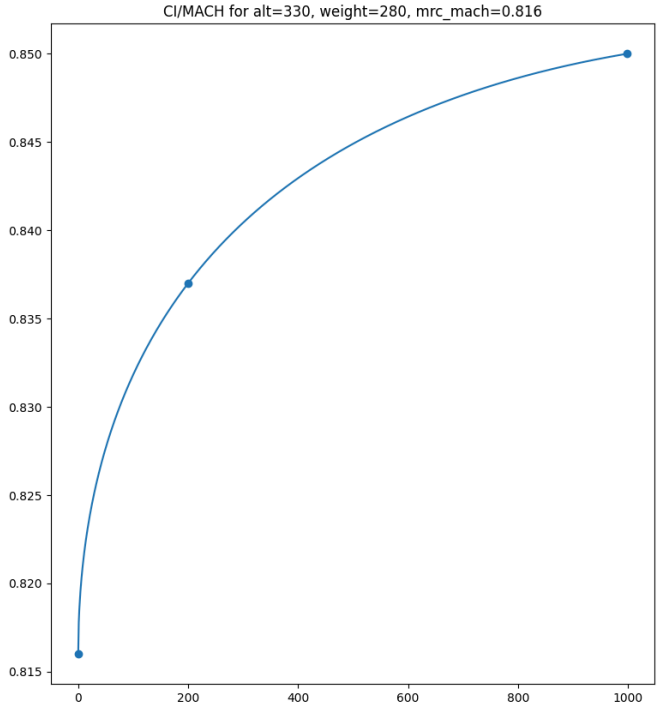




Sanity check: 0.807 MRC and 0.836 LRC

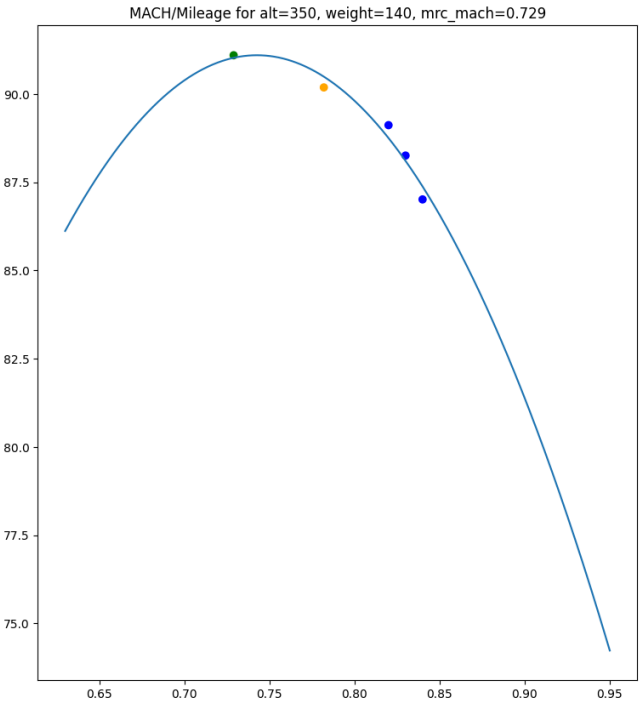
Found: 0.816 MRC and 0.837 LRC

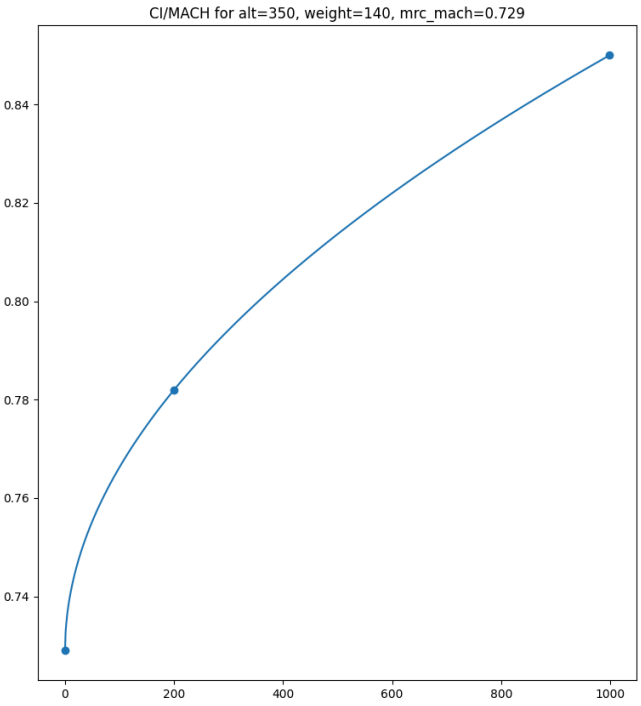




Sanity check: 0.816 MRC and 0.837 LRC

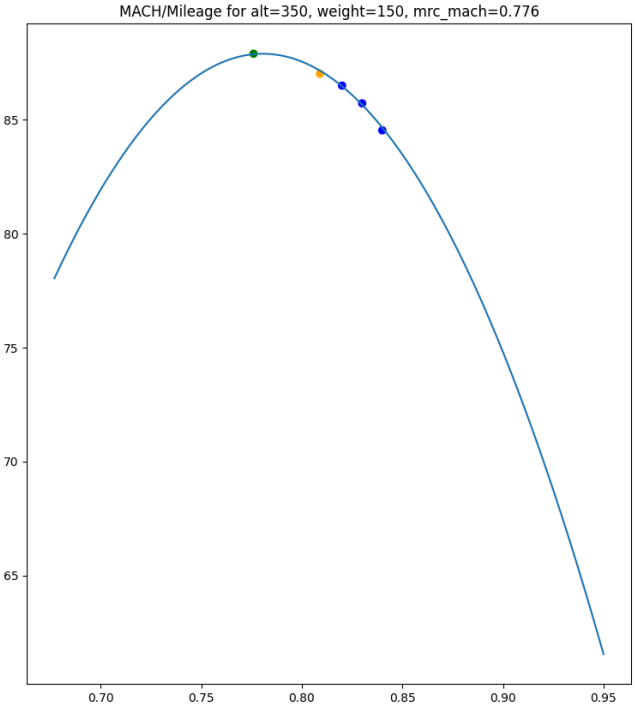
Found: 0.729 MRC and 0.782 LRC

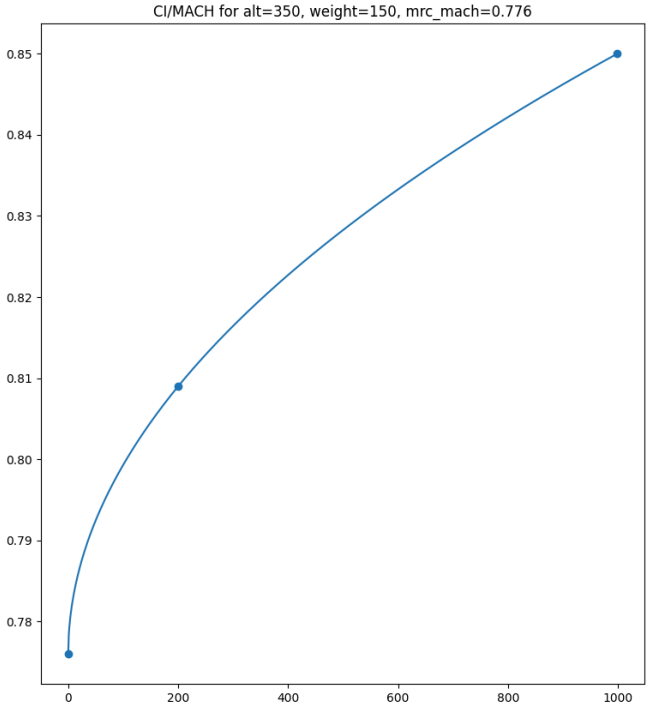




Sanity check: 0.729 MRC and 0.782 LRC

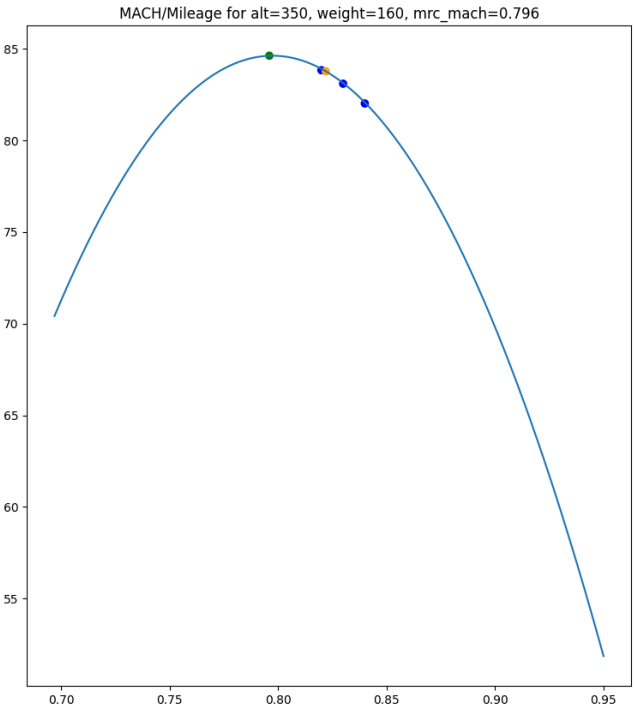
Found: 0.776 MRC and 0.809 LRC

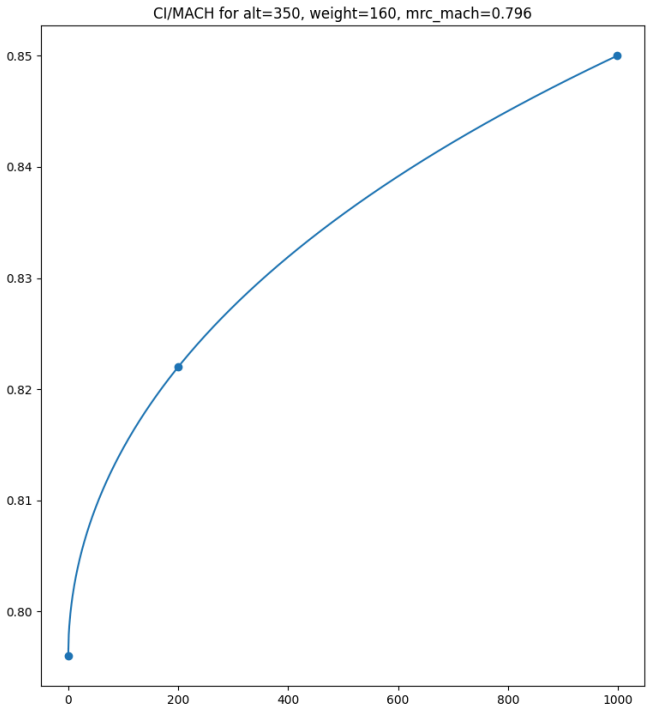




Sanity check: 0.776 MRC and 0.809 LRC

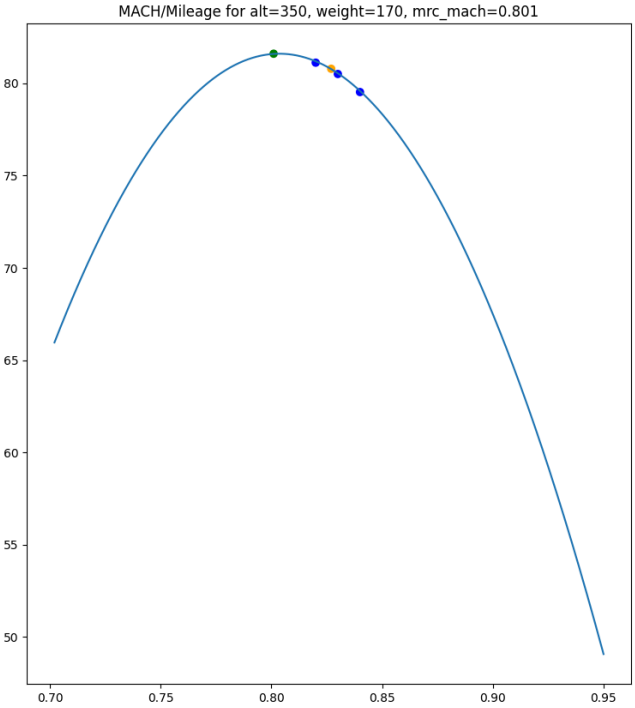
Found: 0.796 MRC and 0.822 LRC

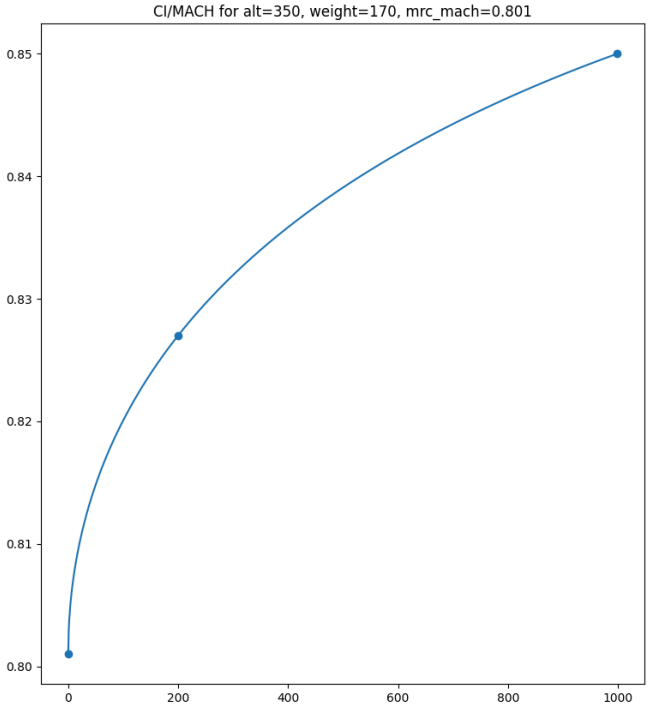




Sanity check: 0.796 MRC and 0.822 LRC

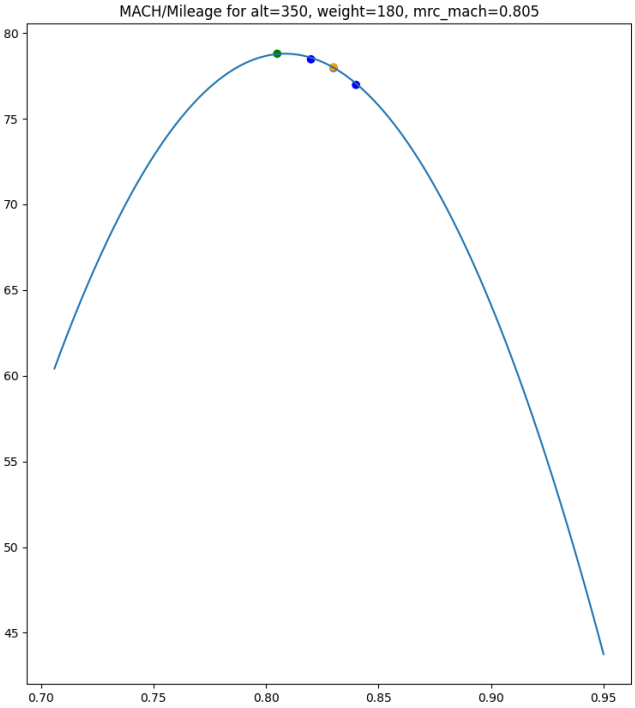
Found: 0.801 MRC and 0.827 LRC

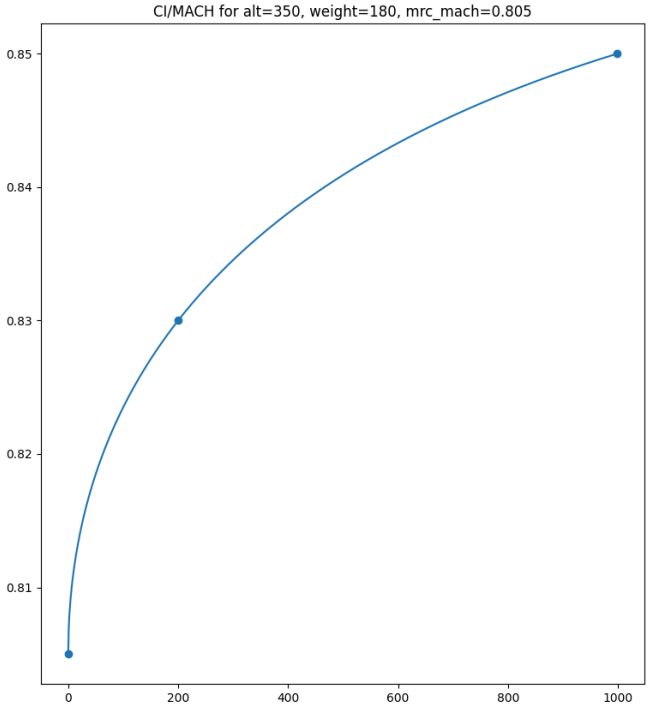




Sanity check: 0.801 MRC and 0.827 LRC

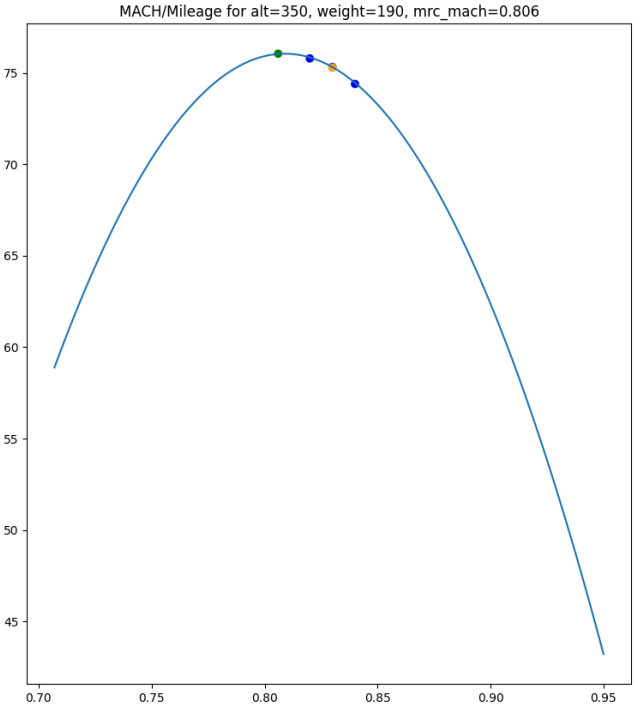
Found: 0.805 MRC and 0.83 LRC

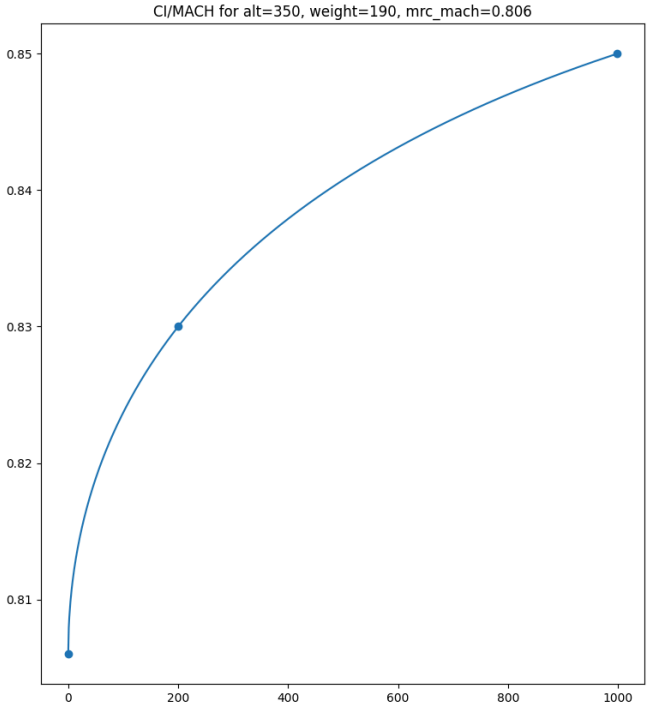




Sanity check: 0.805 MRC and 0.83 LRC

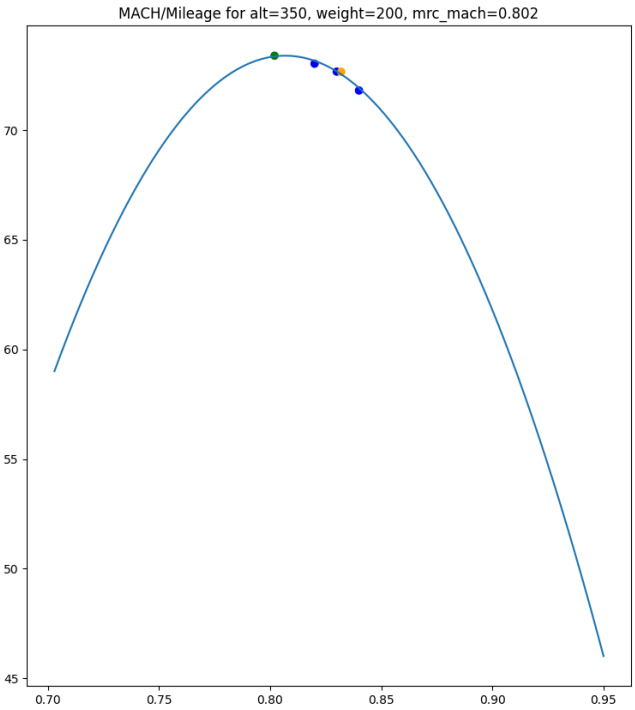
Found: 0.806 MRC and 0.83 LRC

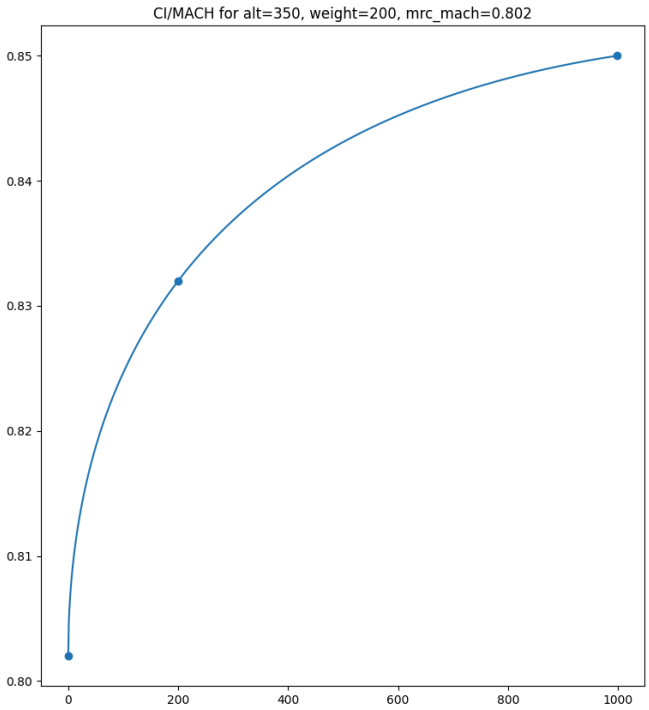




Sanity check: 0.806 MRC and 0.83 LRC

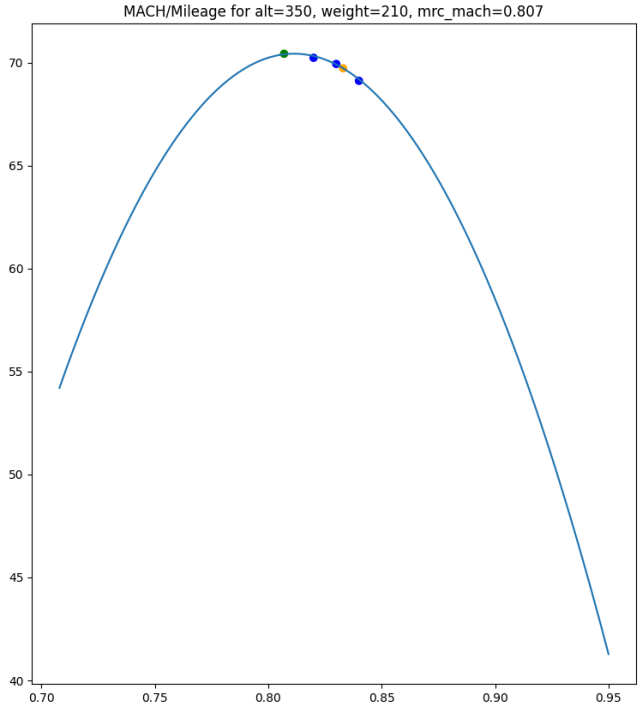
Found: 0.802 MRC and 0.832 LRC

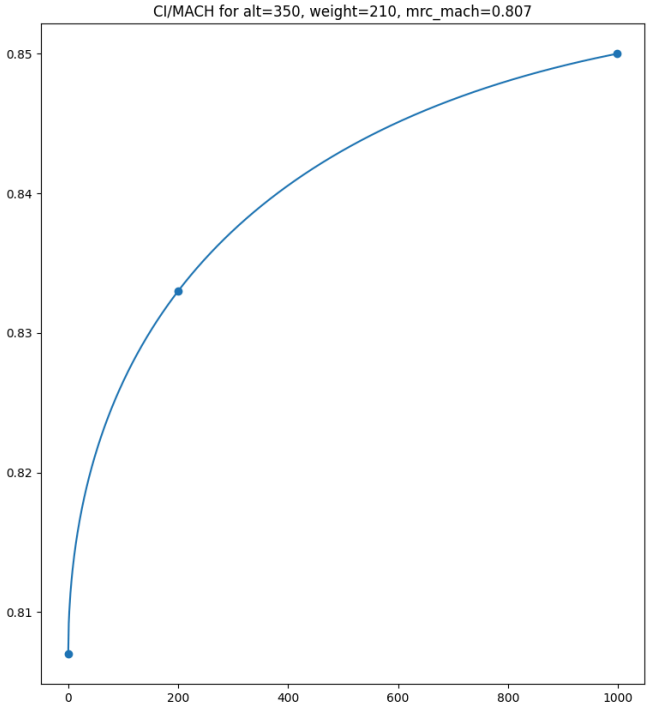




Sanity check: 0.802 MRC and 0.832 LRC

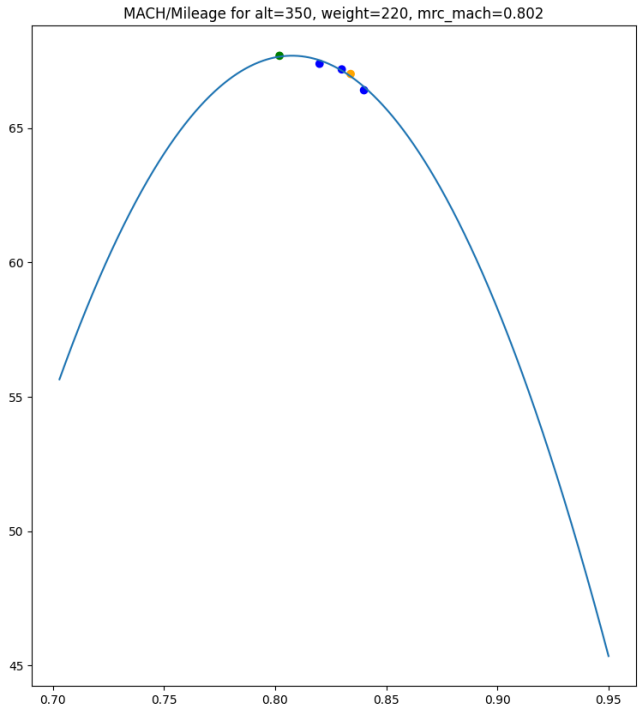
Found: 0.807 MRC and 0.833 LRC

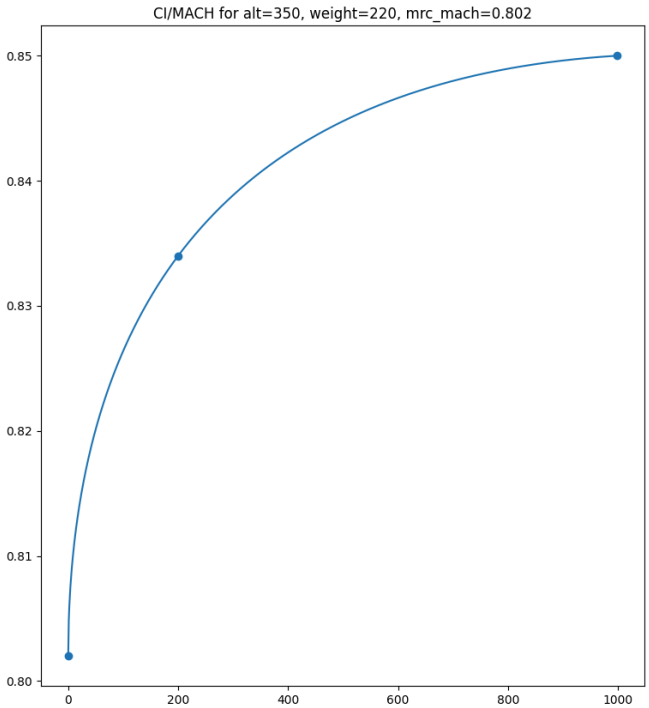




Sanity check: 0.807 MRC and 0.833 LRC

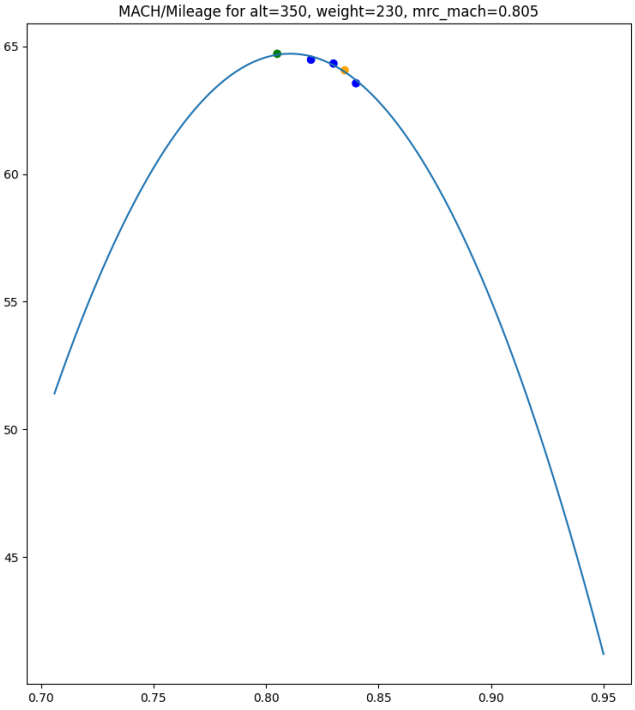
Found: 0.802 MRC and 0.834 LRC

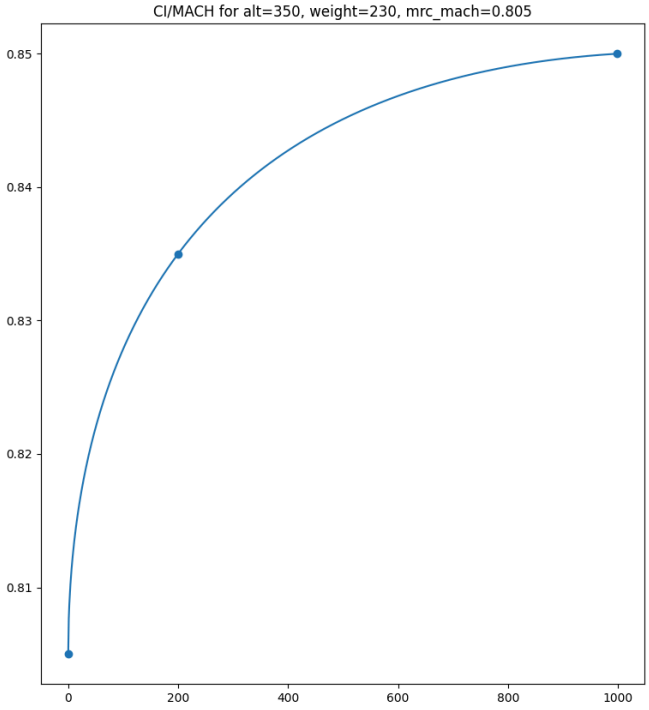




Sanity check: 0.802 MRC and 0.834 LRC

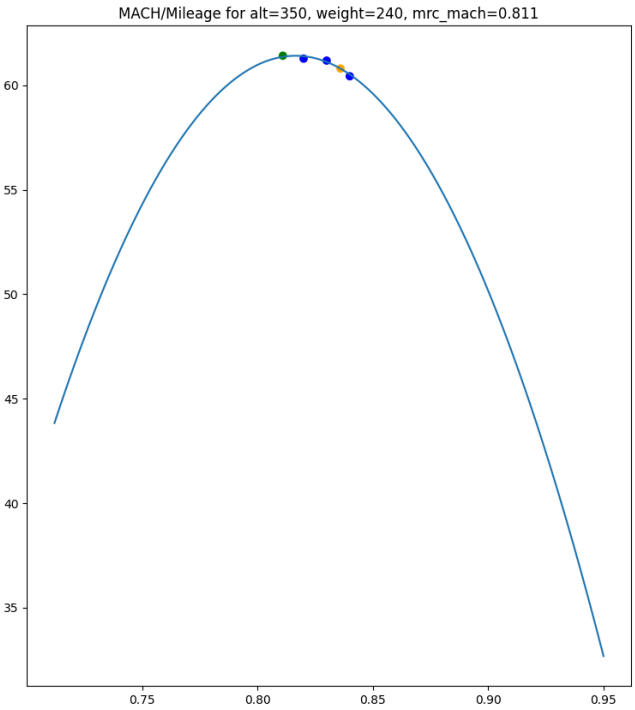
Found: 0.805 MRC and 0.835 LRC

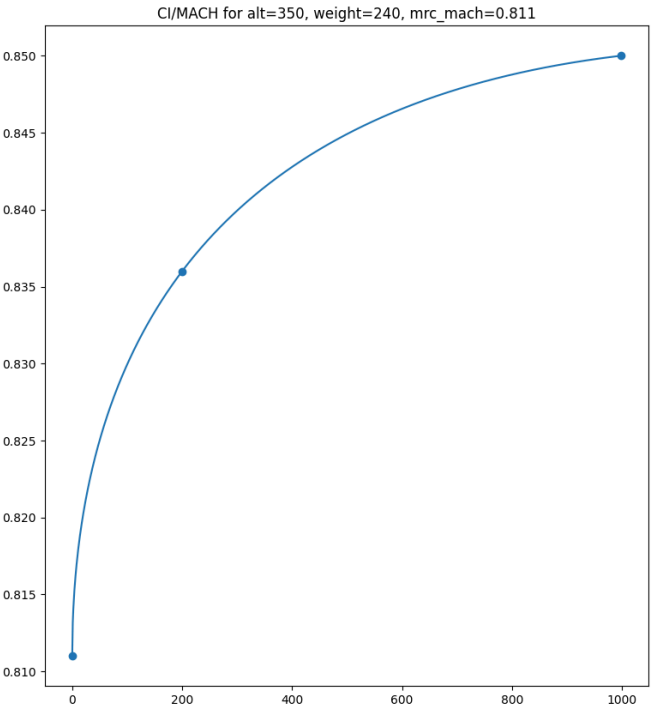




Sanity check: 0.805 MRC and 0.835 LRC

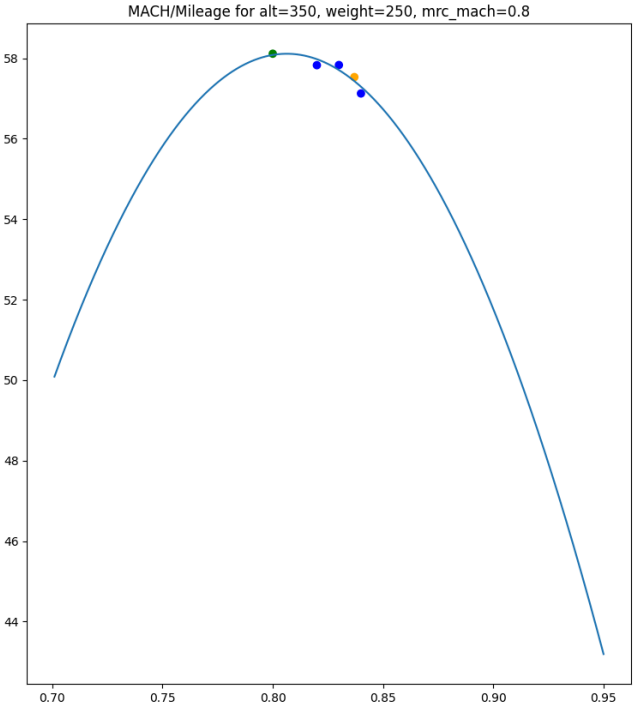
Found: 0.811 MRC and 0.836 LRC

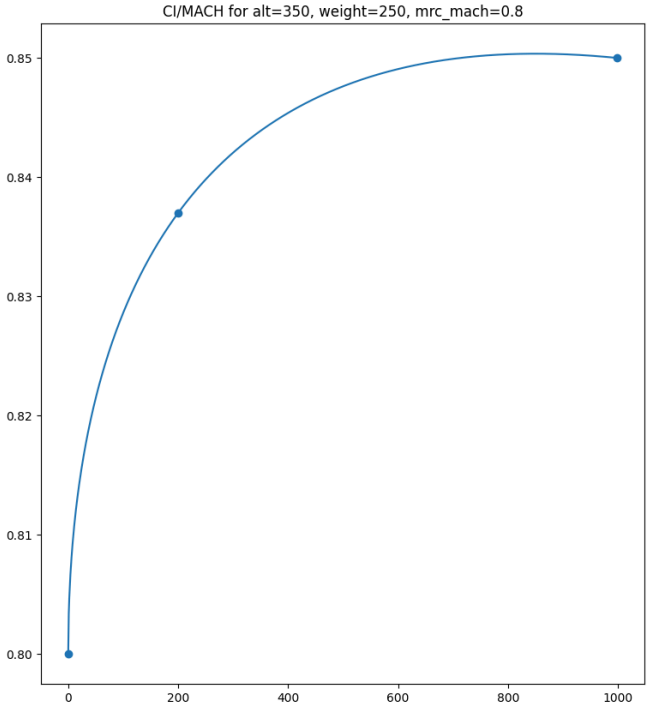




Sanity check: 0.811 MRC and 0.836 LRC

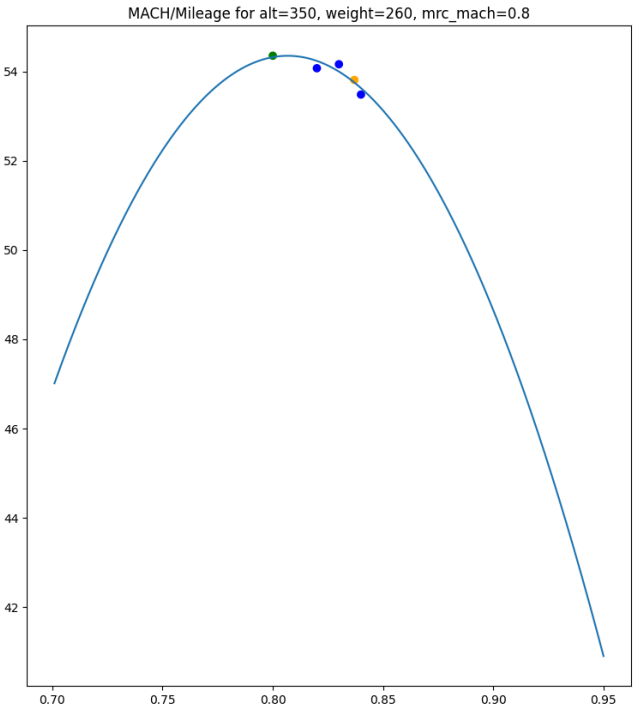
Found: 0.8 MRC and 0.837 LRC

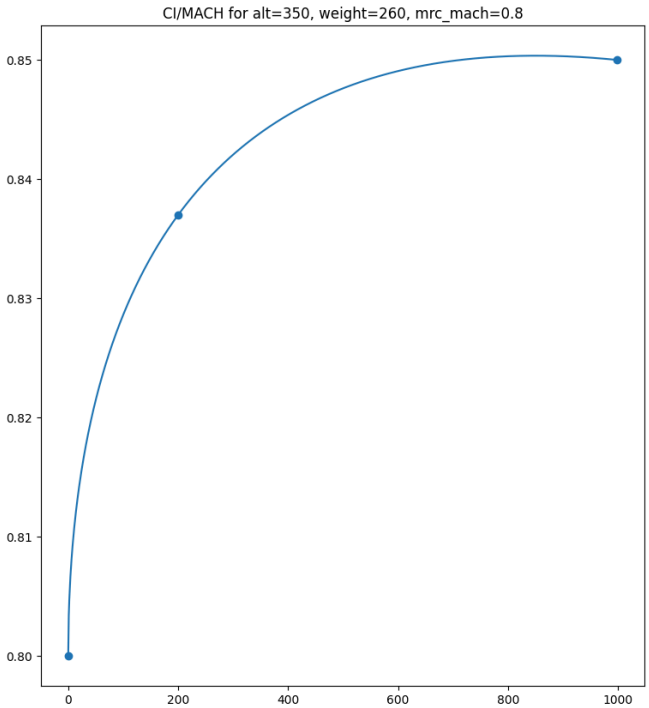




Sanity check: 0.8 MRC and 0.837 LRC

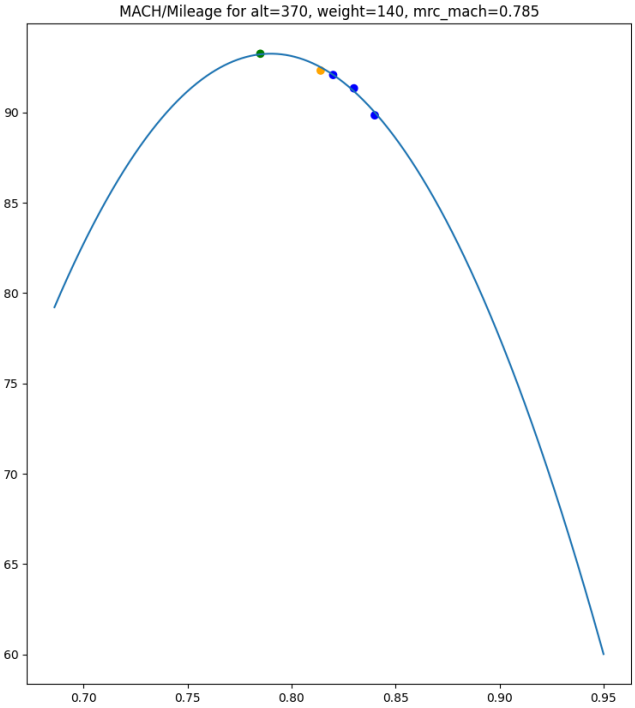
Found: 0.8 MRC and 0.837 LRC

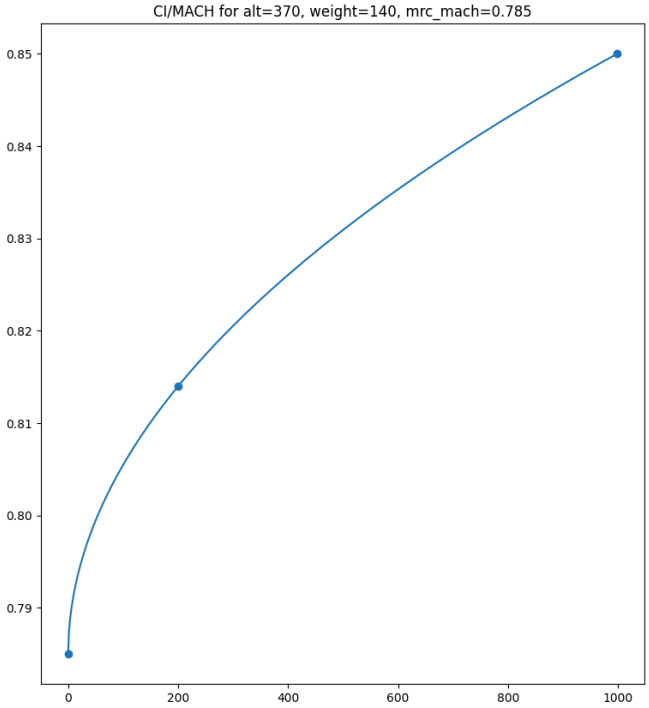




Sanity check: 0.8 MRC and 0.837 LRC

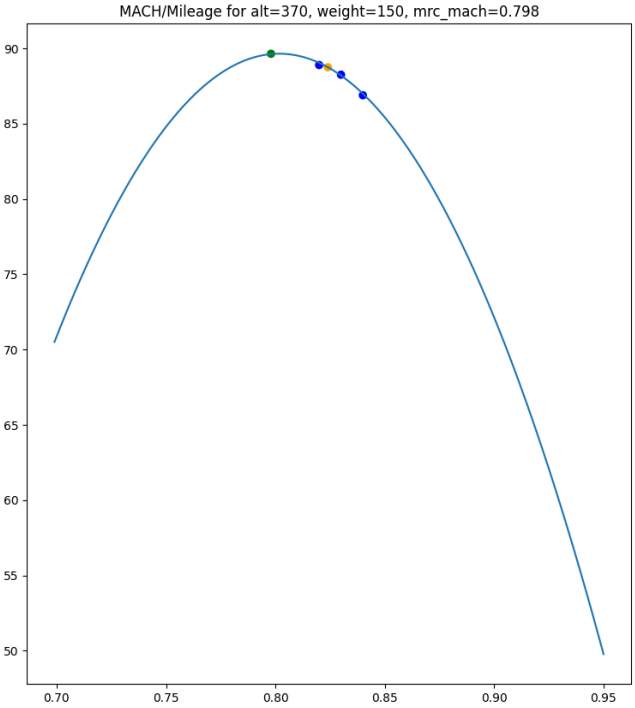
Found: 0.785 MRC and 0.814 LRC

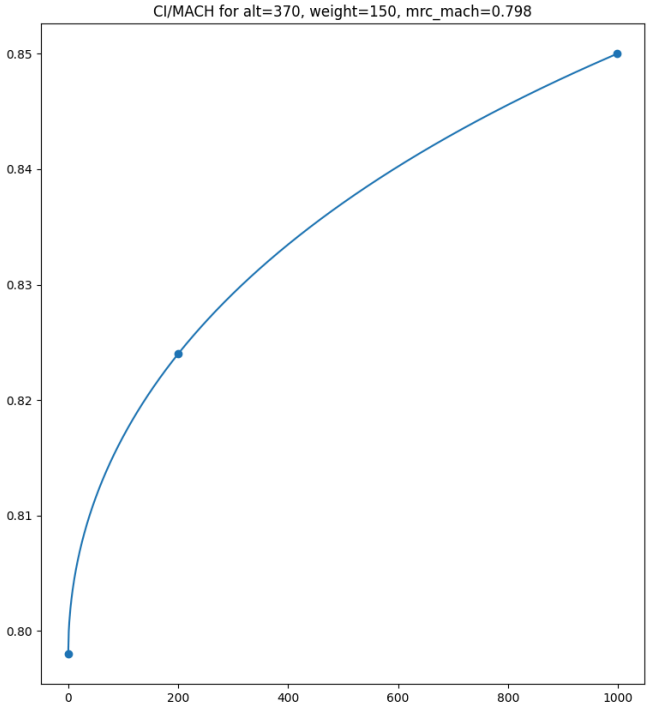




Sanity check: 0.785 MRC and 0.814 LRC

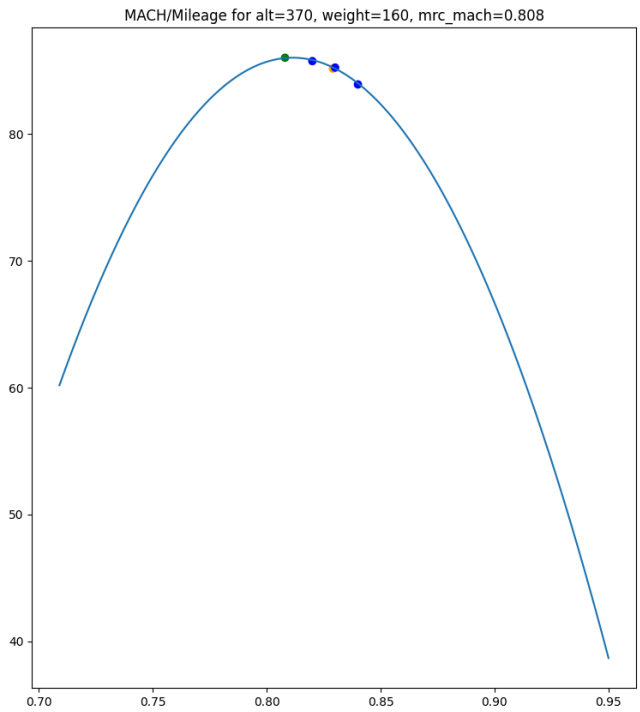
Found: 0.798 MRC and 0.824 LRC

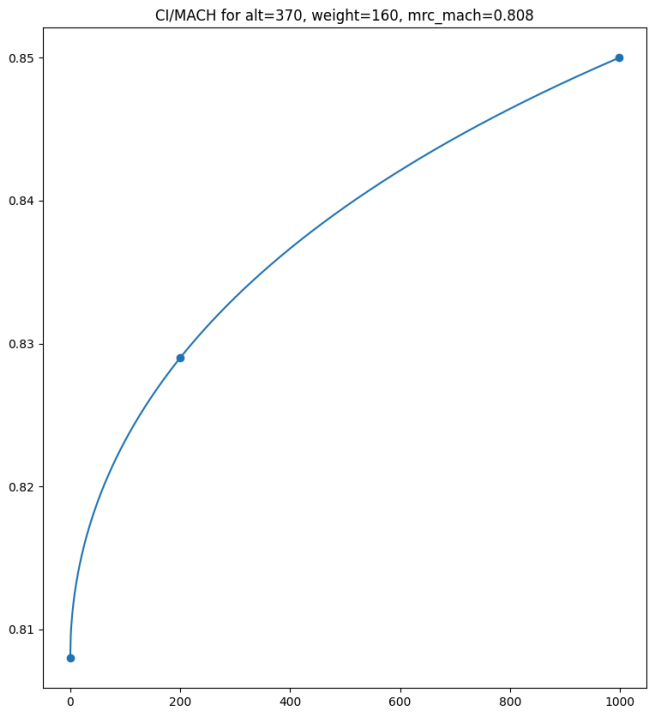




Sanity check: 0.798 MRC and 0.824 LRC

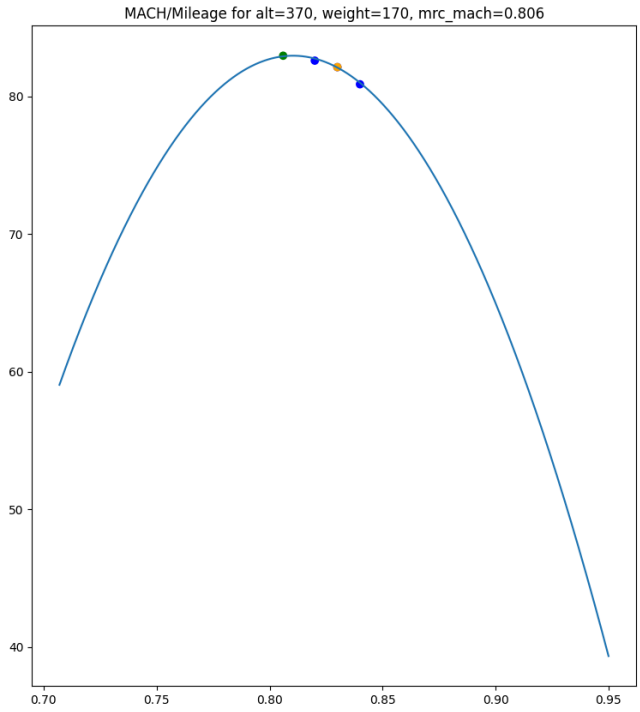
Found: 0.808 MRC and 0.829 LRC

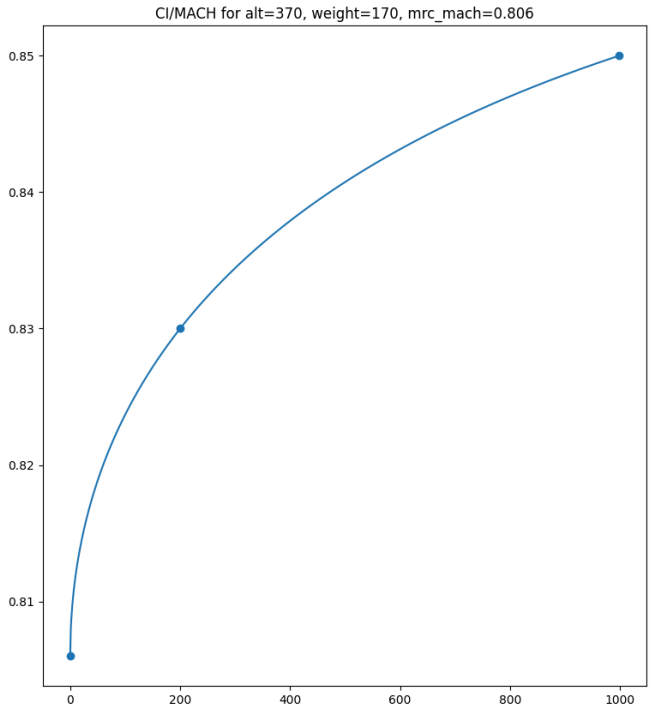




Sanity check: 0.808 MRC and 0.829 LRC

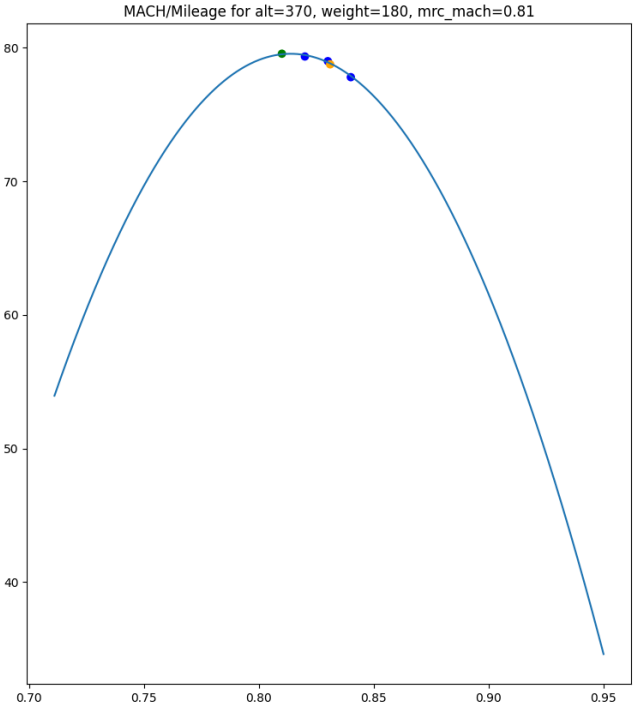
Found: 0.806 MRC and 0.83 LRC

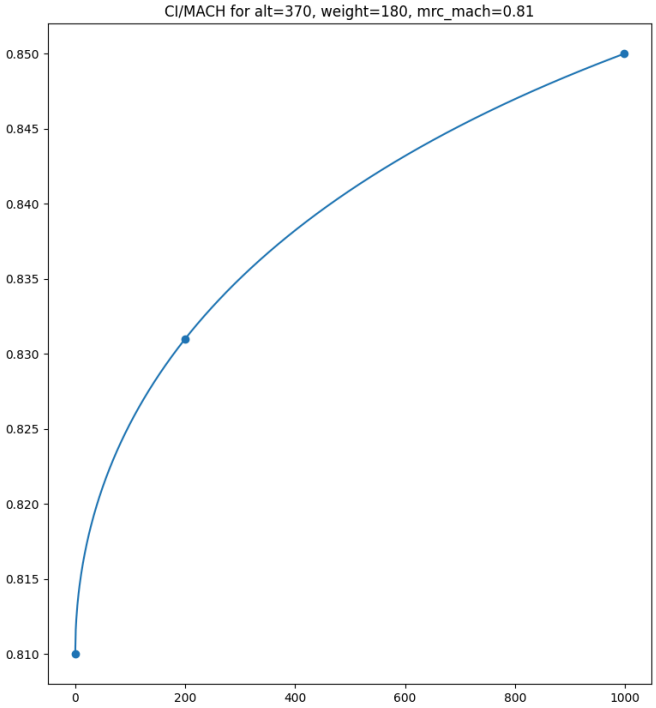




Sanity check: 0.806 MRC and 0.83 LRC

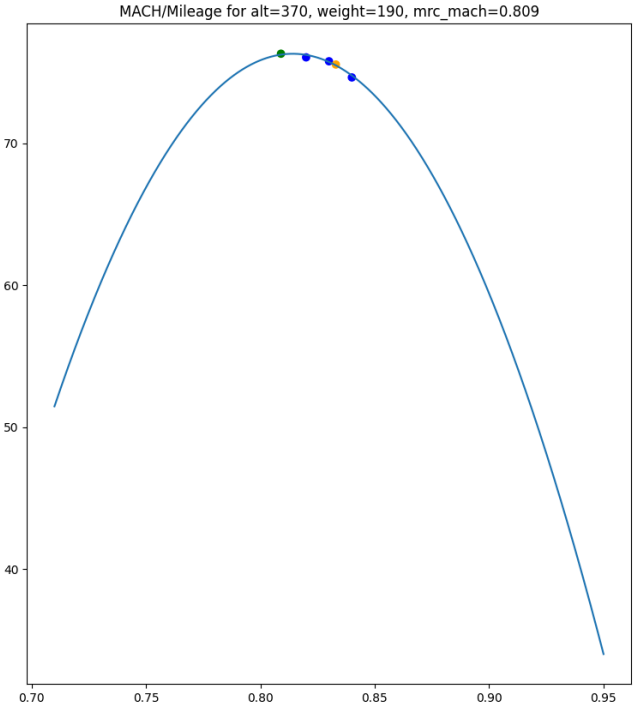
Found: 0.81 MRC and 0.831 LRC

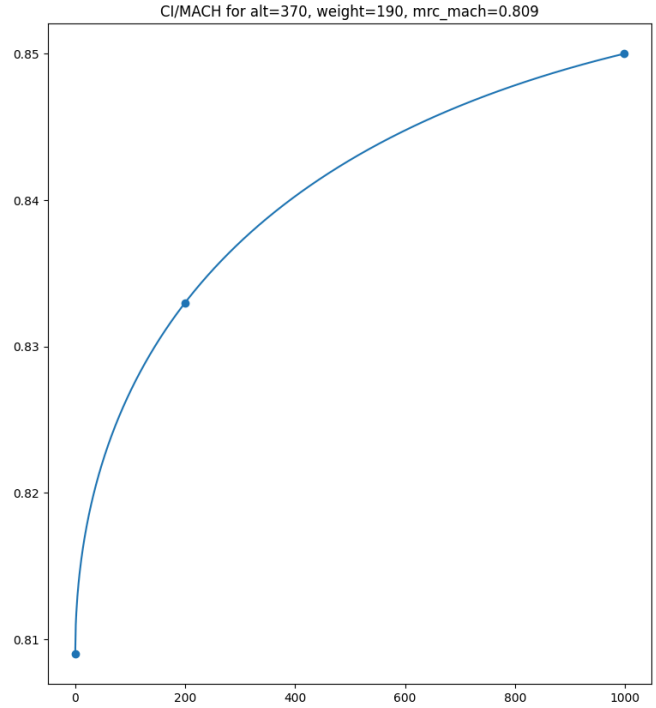




Sanity check: 0.81 MRC and 0.831 LRC

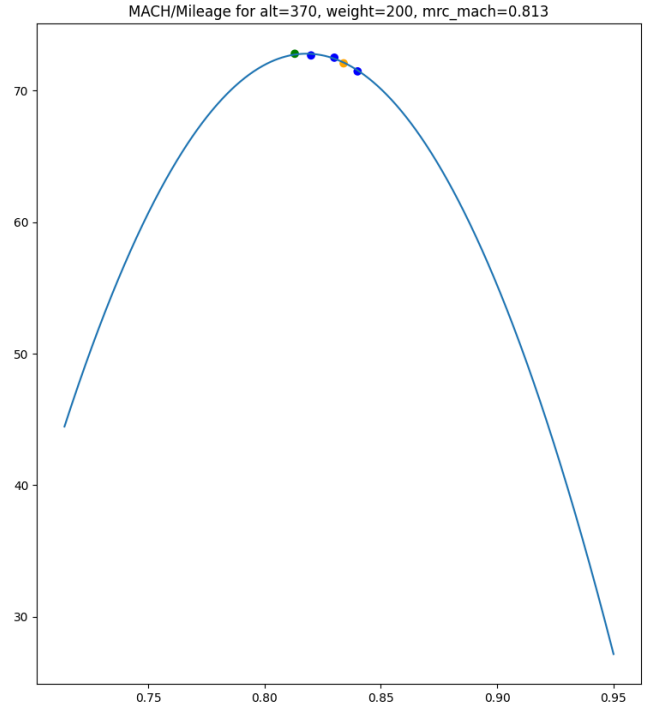
Found: 0.809 MRC and 0.833 LRC

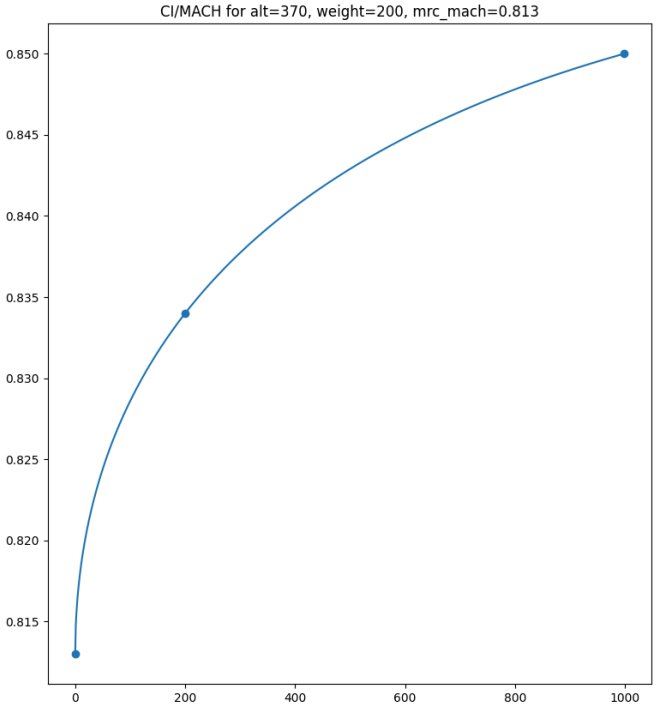




Sanity check: 0.809 MRC and 0.833 LRC

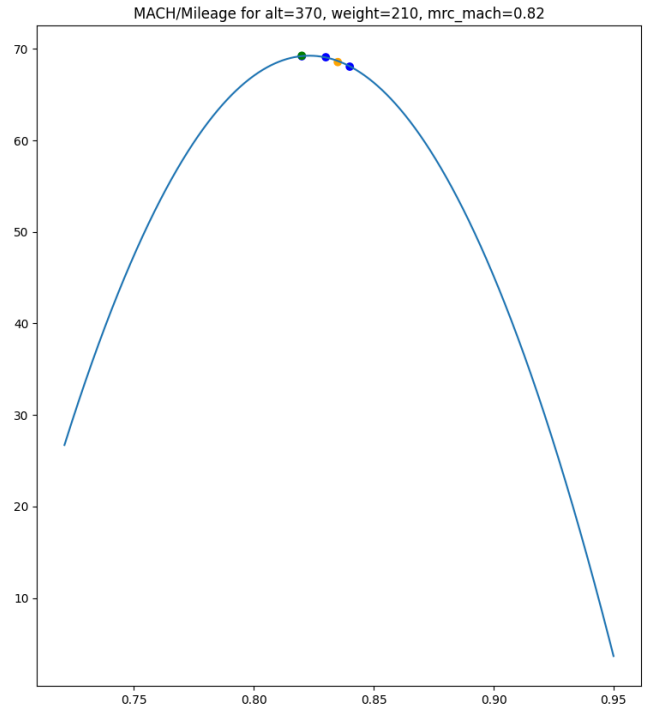
Found: 0.813 MRC and 0.834 LRC

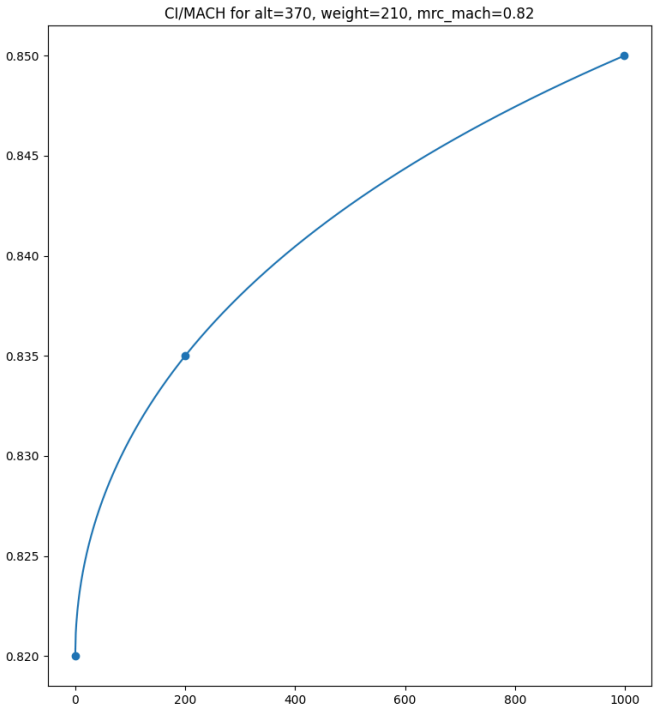




Sanity check: 0.813 MRC and 0.834 LRC

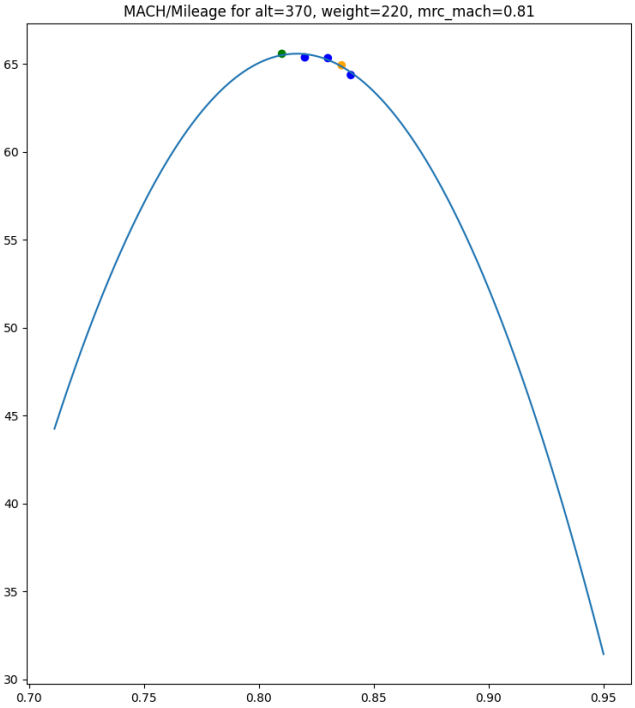
Found: 0.82 MRC and 0.835 LRC

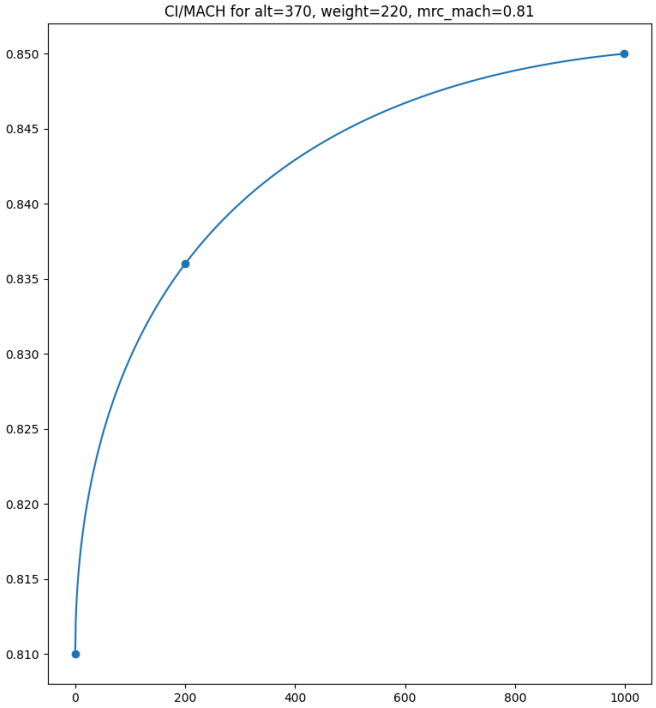




Sanity check: 0.82 MRC and 0.835 LRC

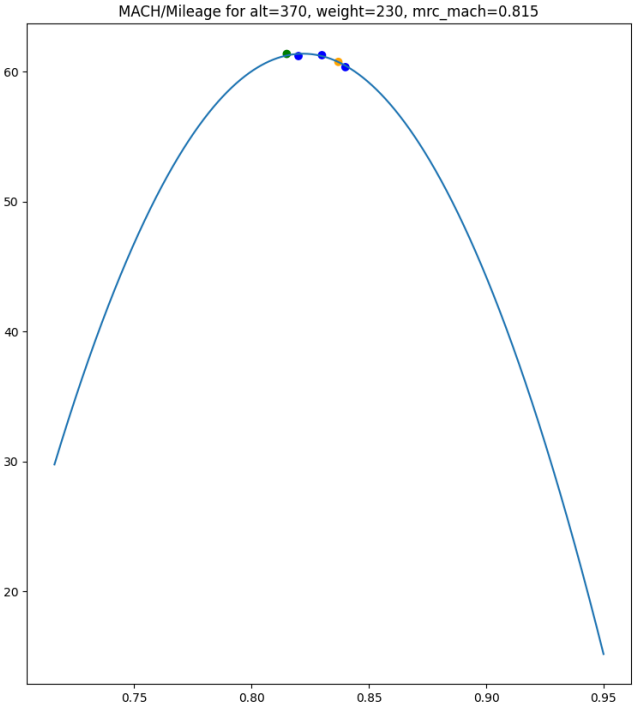
Found: 0.81 MRC and 0.836 LRC

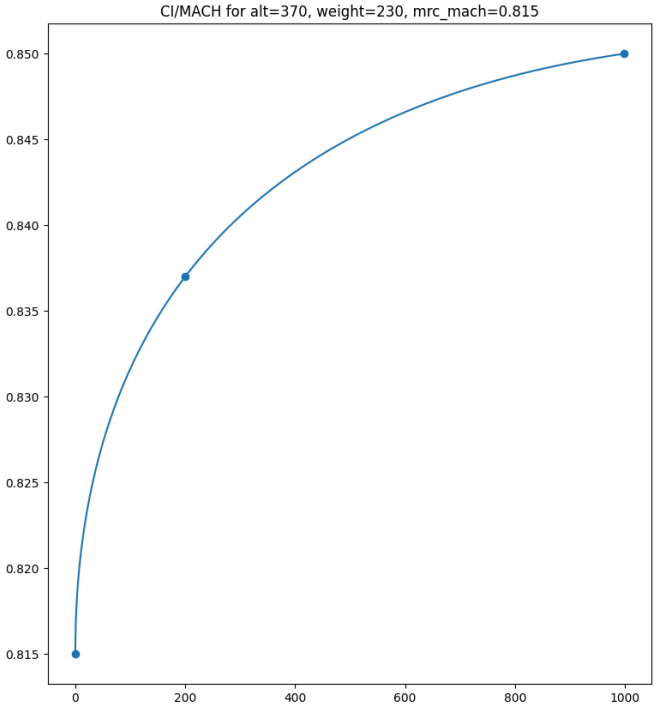




Sanity check: 0.81 MRC and 0.836 LRC

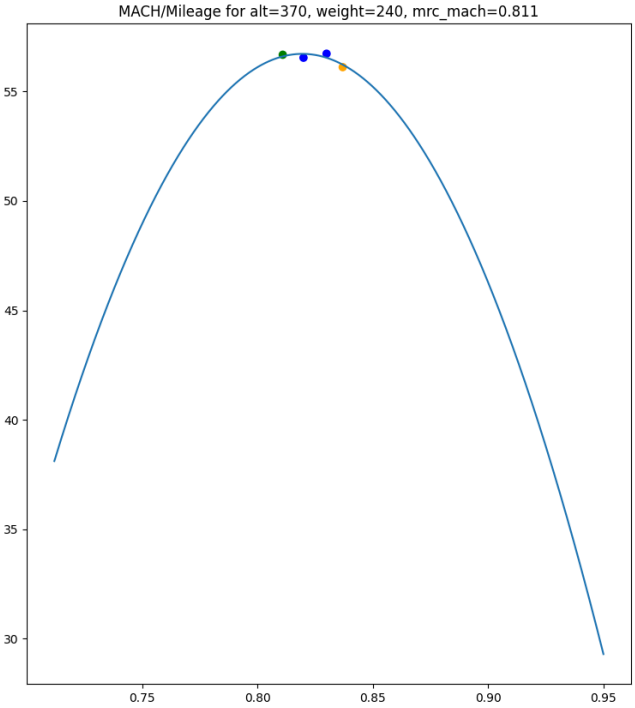
Found: 0.815 MRC and 0.837 LRC

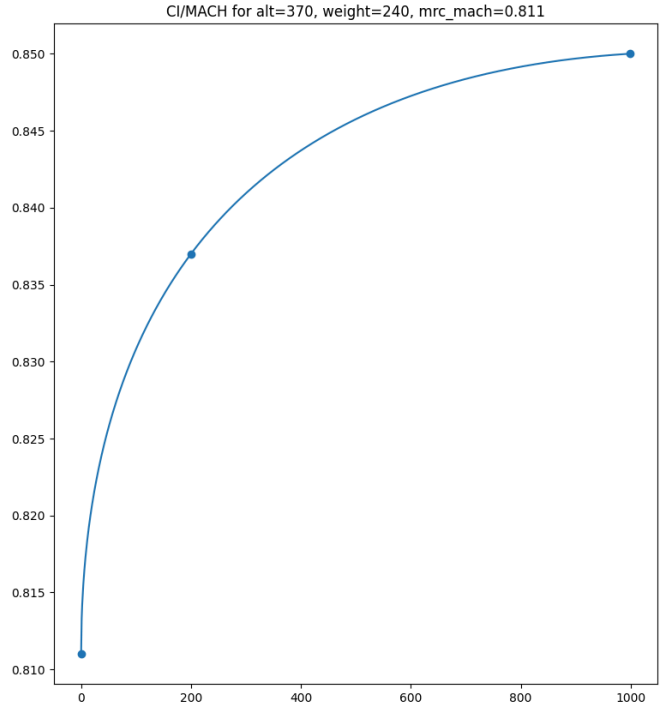




Sanity check: 0.815 MRC and 0.837 LRC

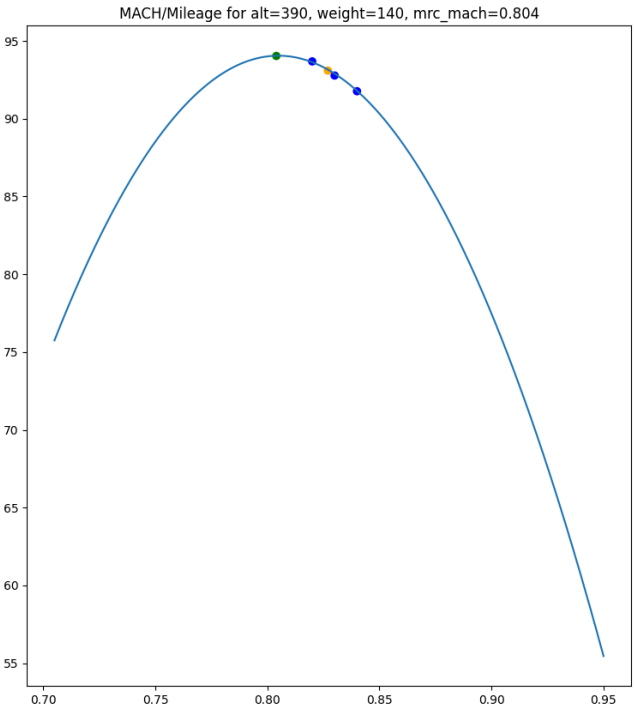
Found: 0.811 MRC and 0.837 LRC

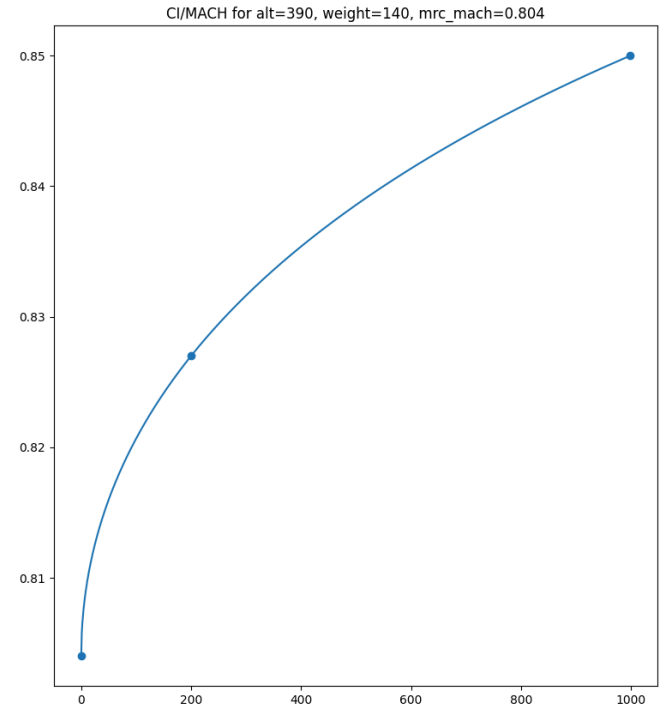




Sanity check: 0.811 MRC and 0.837 LRC

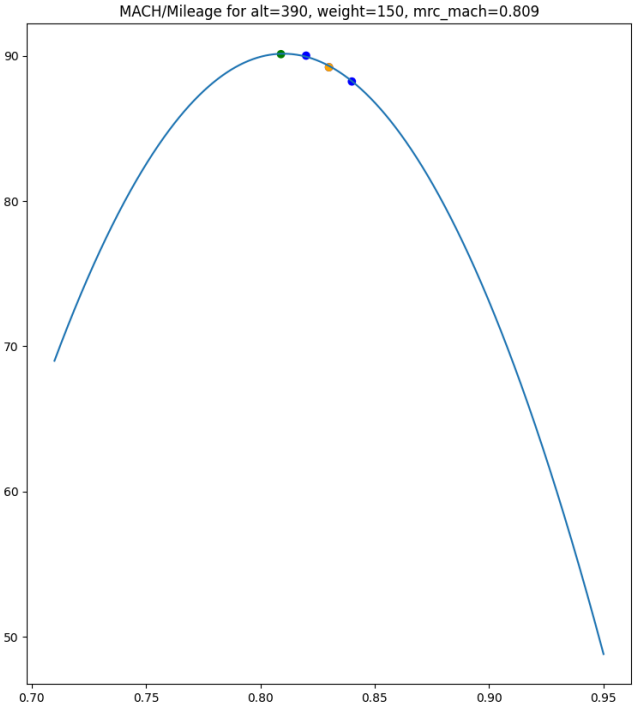
Found: 0.804 MRC and 0.827 LRC

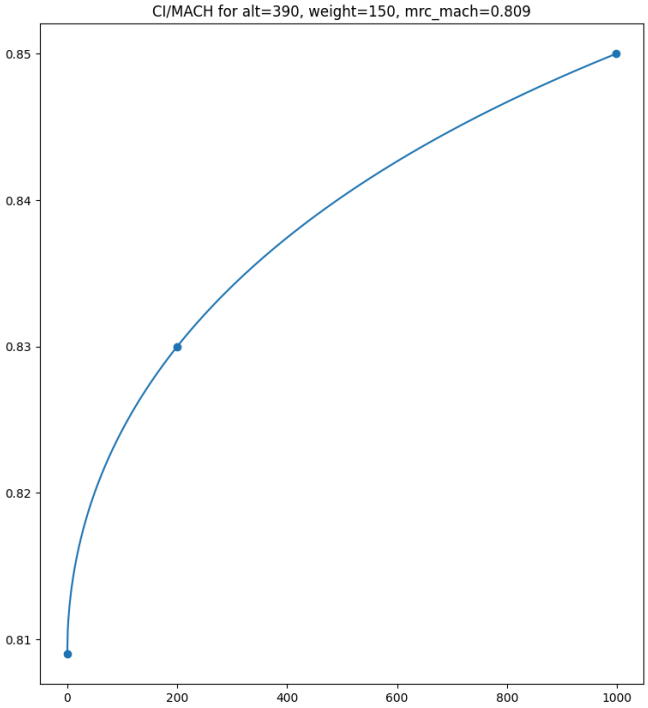




Sanity check: 0.804 MRC and 0.827 LRC

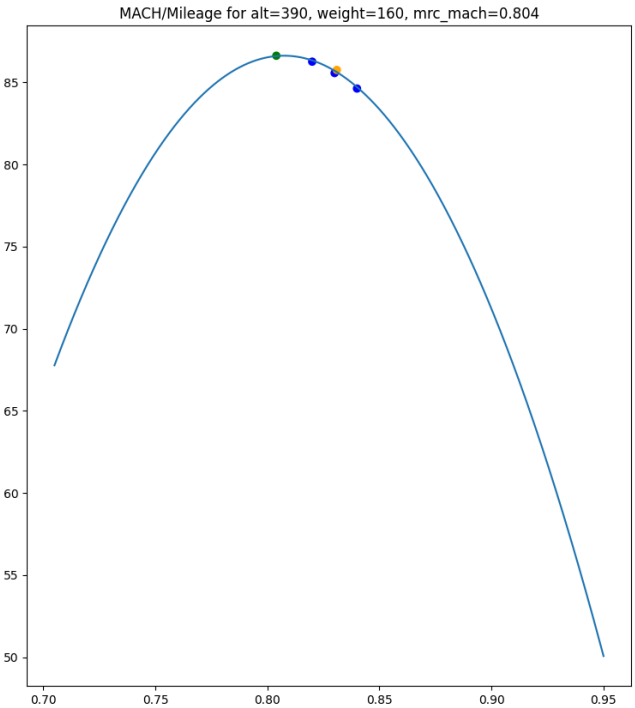
Found: 0.809 MRC and 0.83 LRC

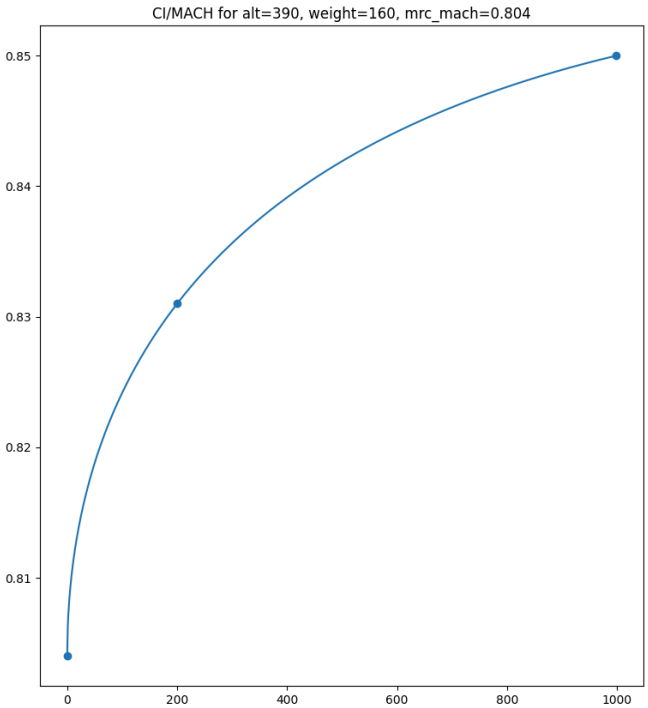




Sanity check: 0.809 MRC and 0.83 LRC

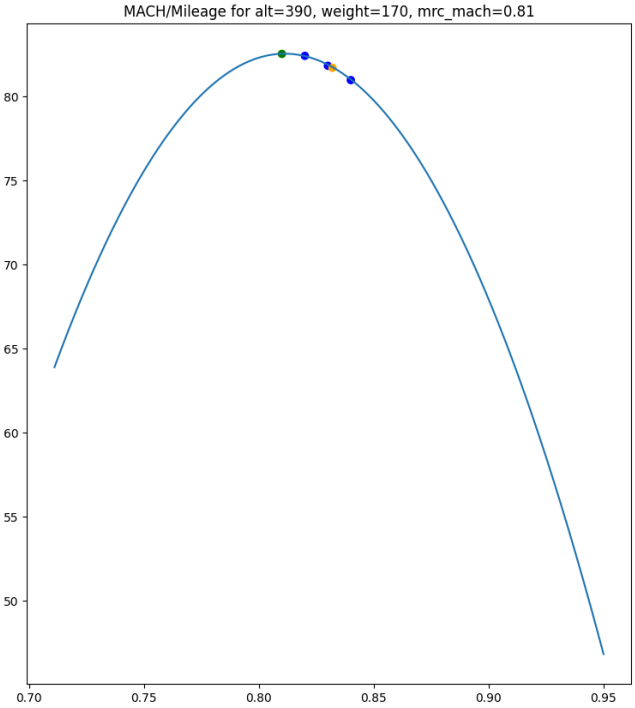
Found: 0.804 MRC and 0.831 LRC

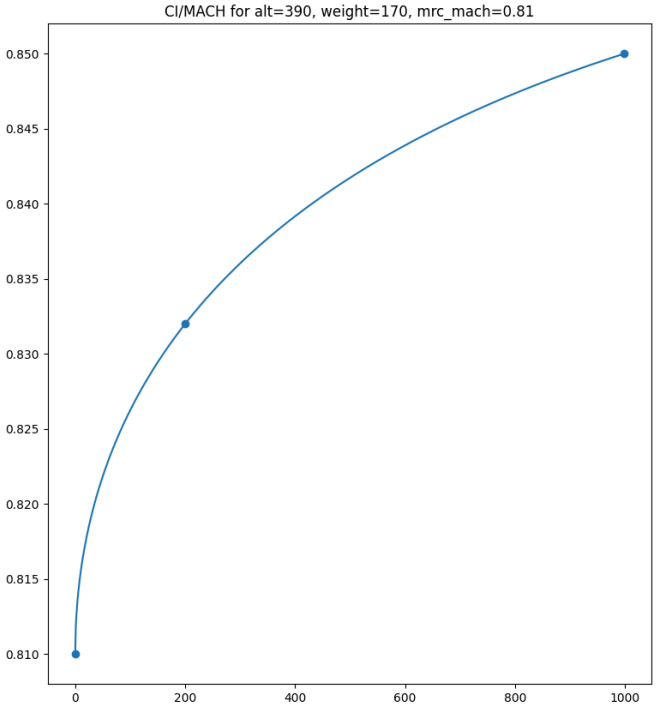




Sanity check: 0.804 MRC and 0.831 LRC

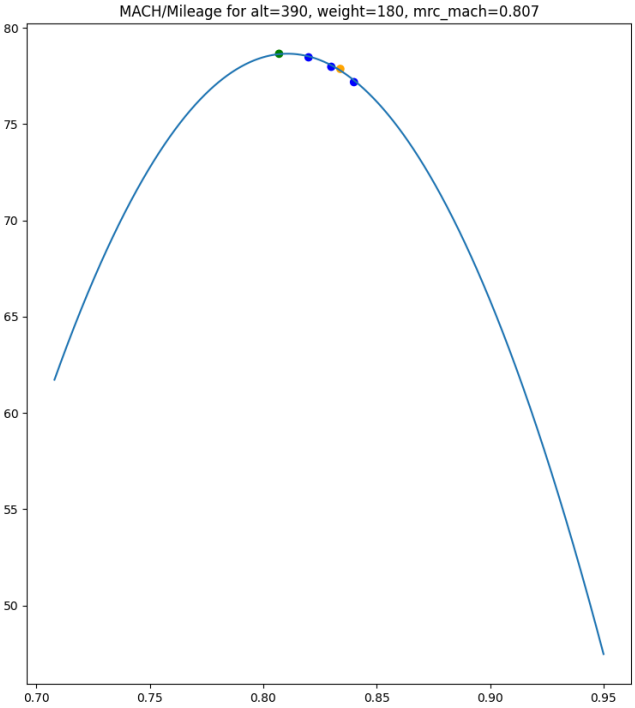
Found: 0.81 MRC and 0.832 LRC

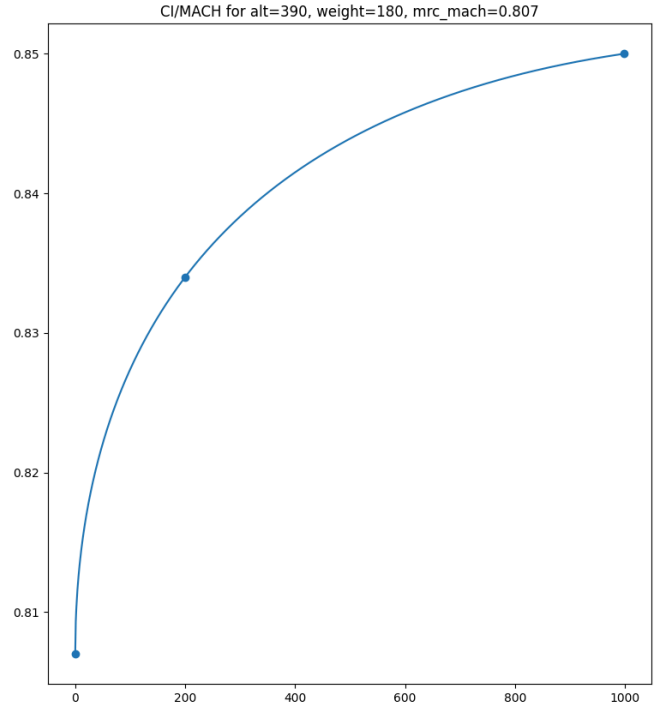




Sanity check: 0.81 MRC and 0.832 LRC

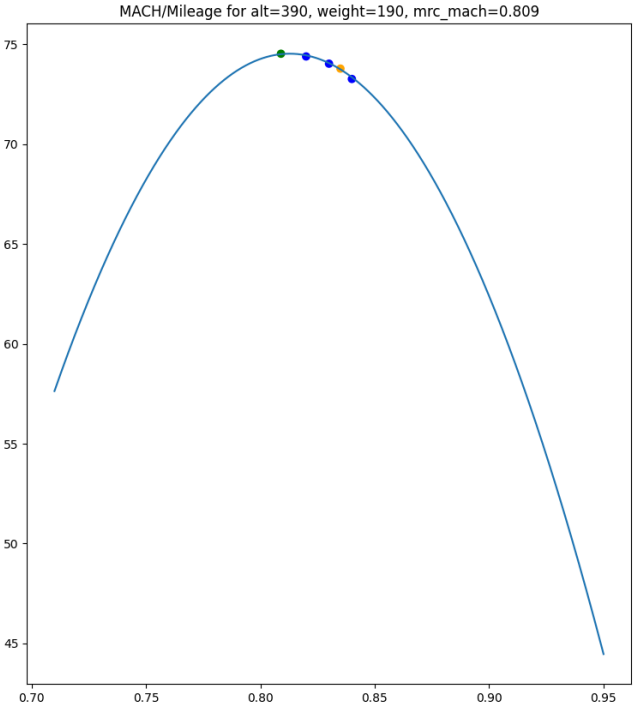
Found: 0.807 MRC and 0.834 LRC

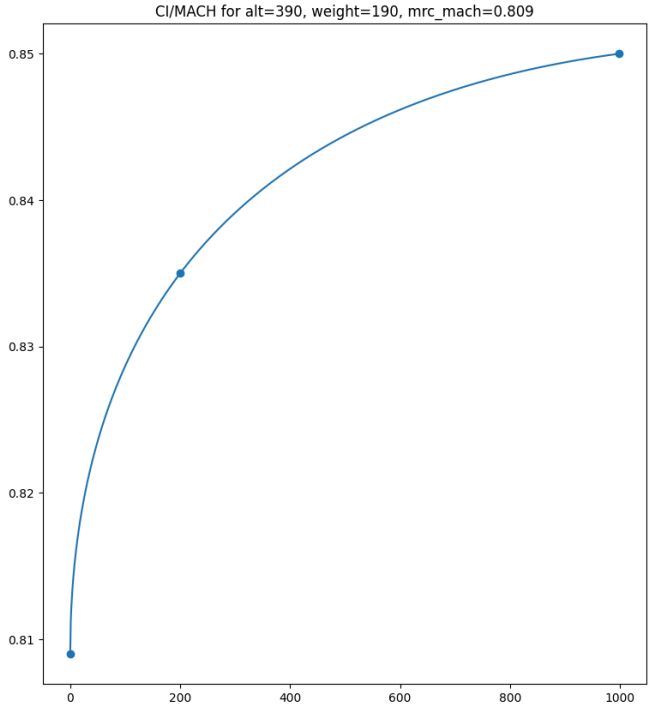




Sanity check: 0.807 MRC and 0.834 LRC

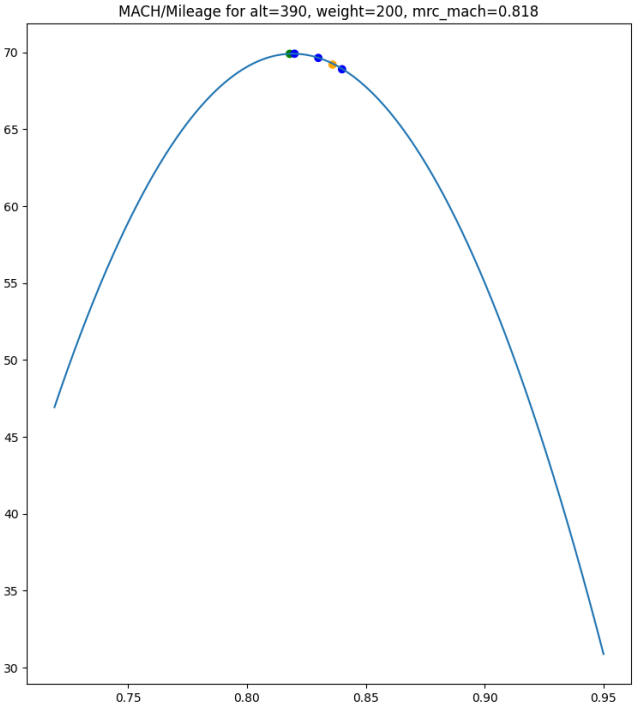
Found: 0.809 MRC and 0.835 LRC

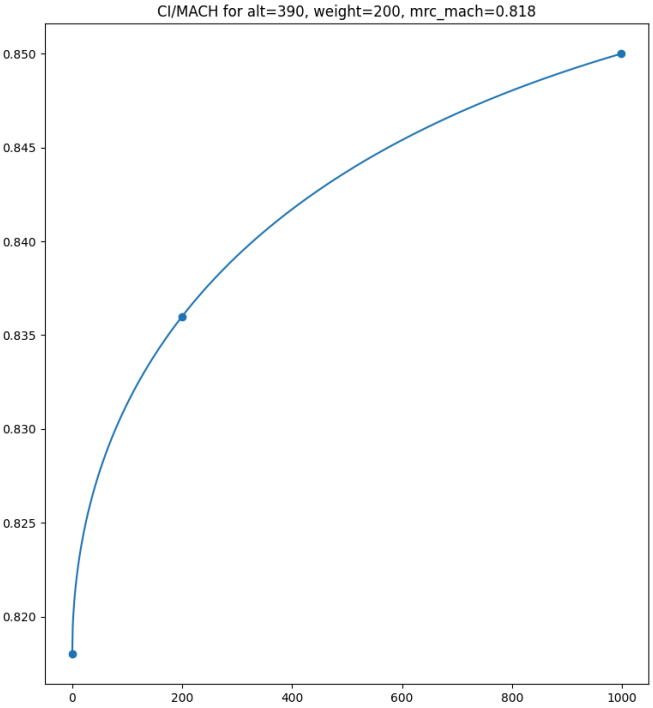




Sanity check: 0.809 MRC and 0.835 LRC

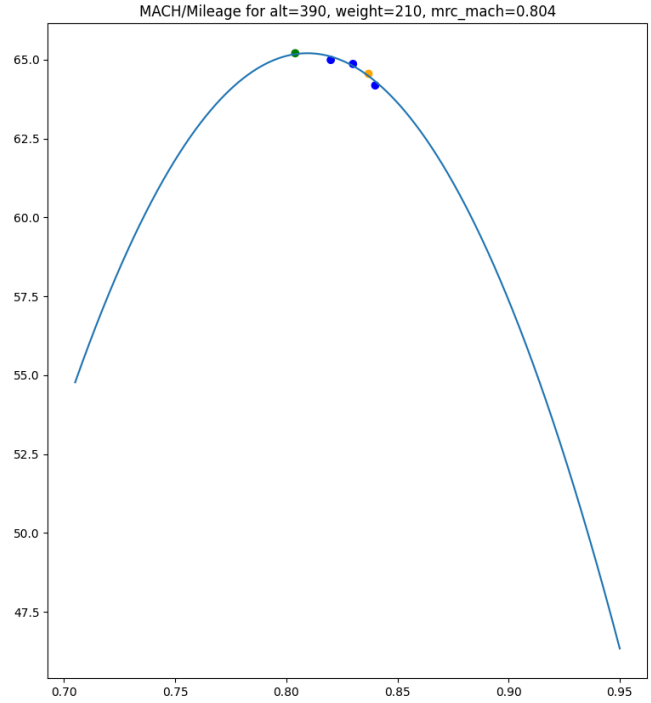
Found: 0.818 MRC and 0.836 LRC

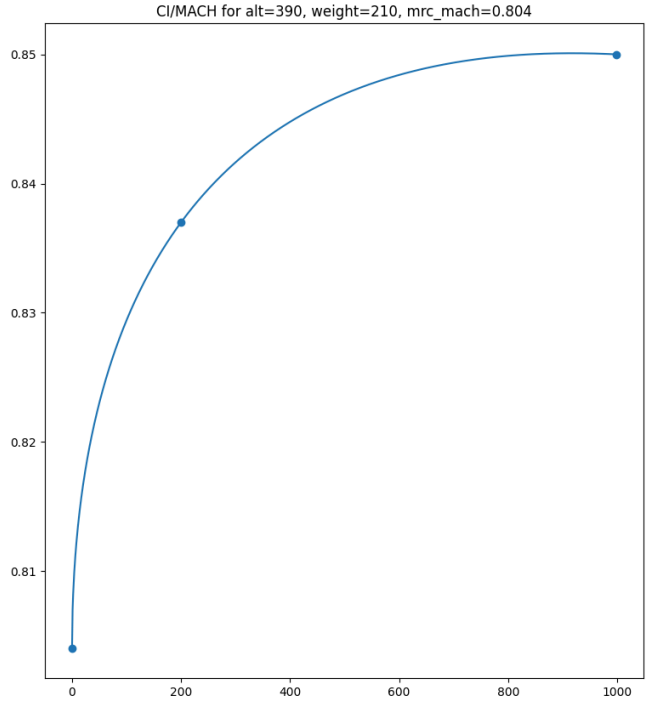




Sanity check: 0.818 MRC and 0.836 LRC

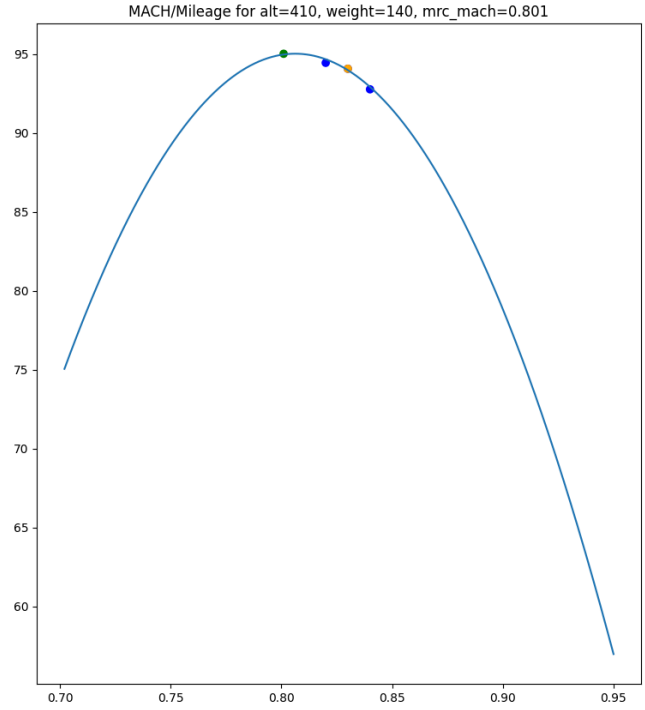
Found: 0.804 MRC and 0.837 LRC

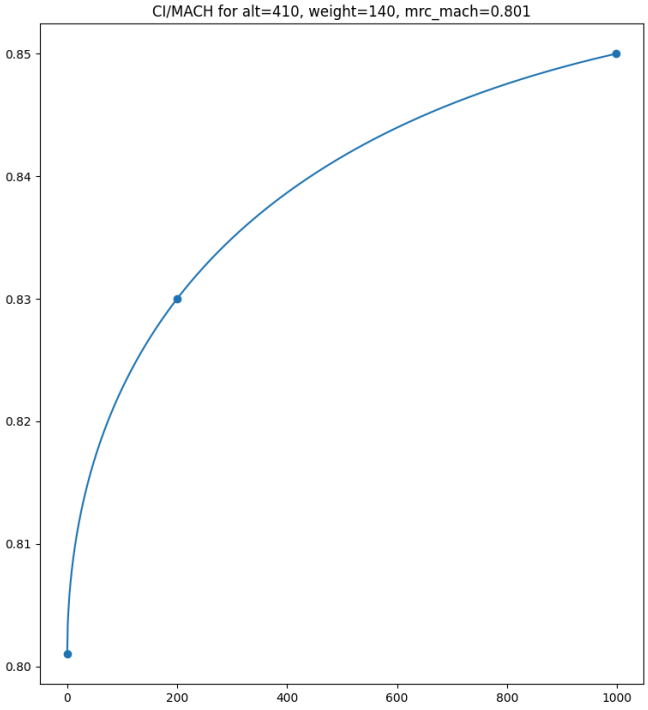




Sanity check: 0.804 MRC and 0.837 LRC

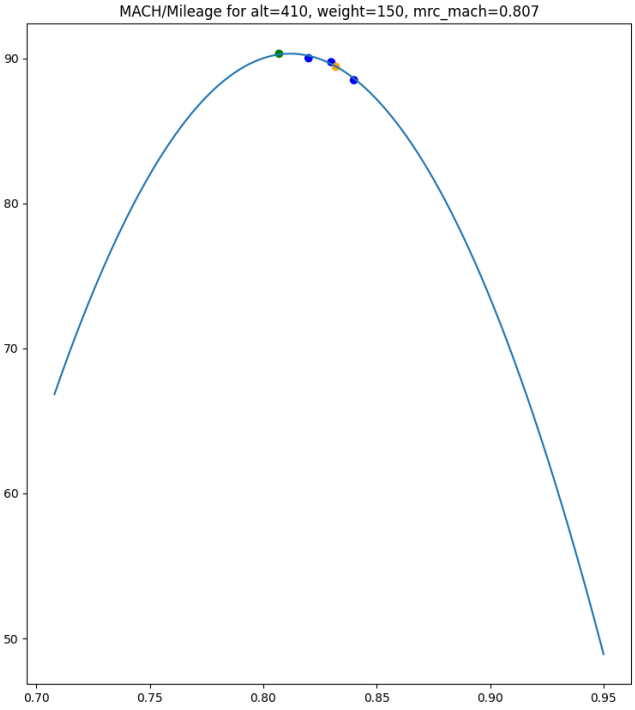
Found: 0.801 MRC and 0.83 LRC

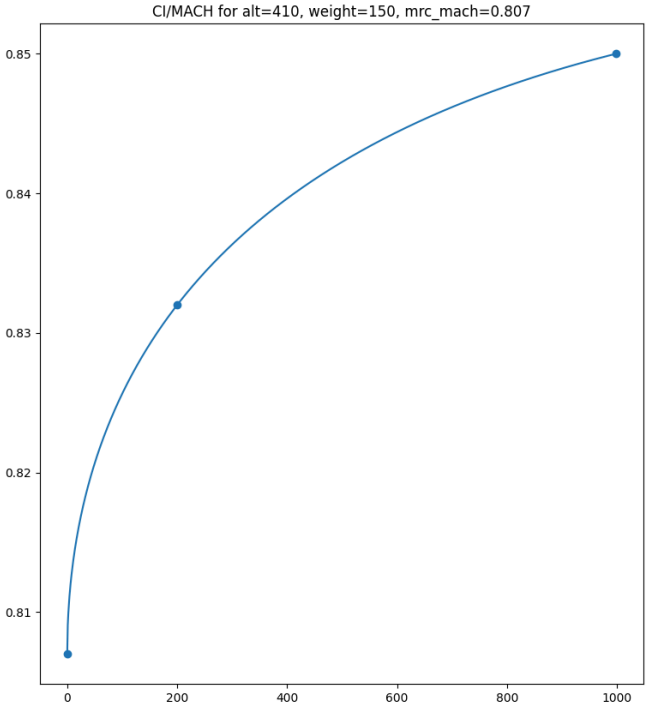




Sanity check: 0.801 MRC and 0.83 LRC

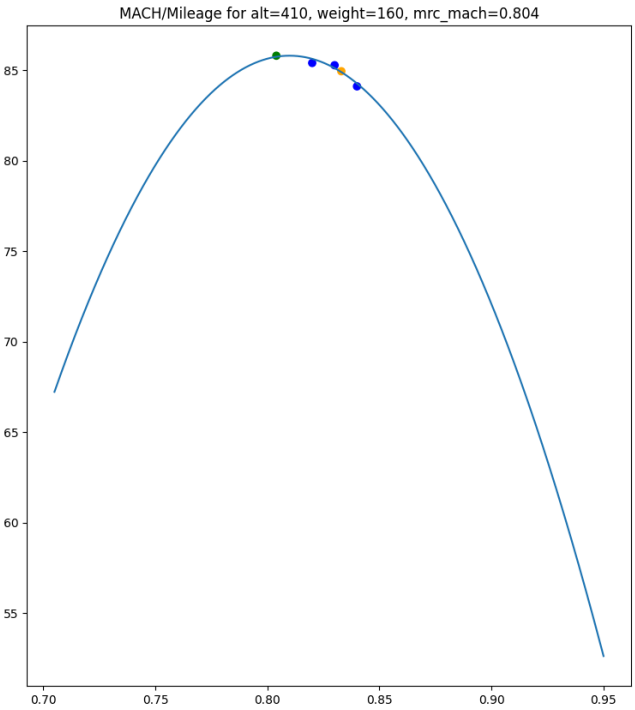
Found: 0.807 MRC and 0.832 LRC

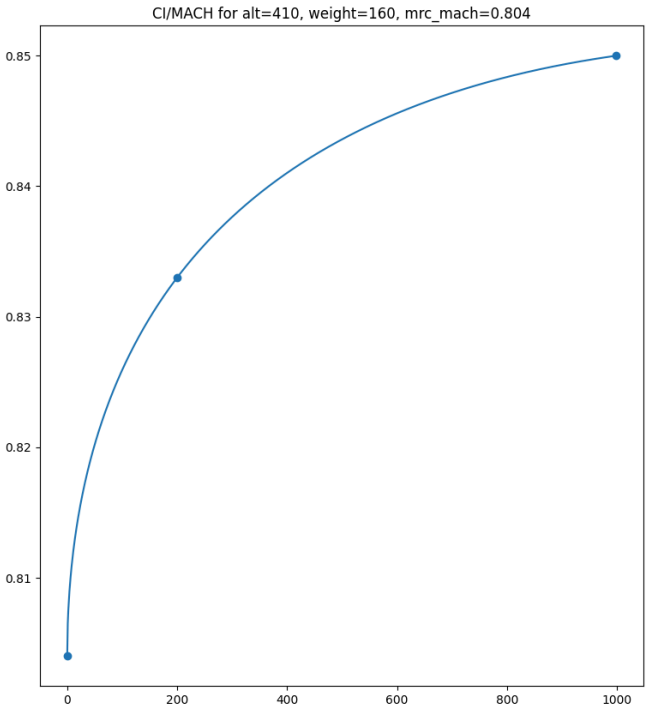




Sanity check: 0.807 MRC and 0.832 LRC

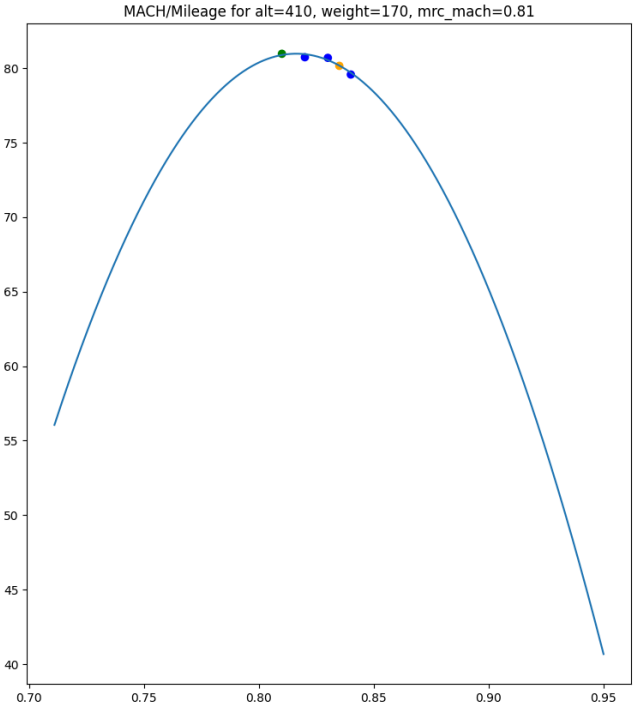
Found: 0.804 MRC and 0.833 LRC

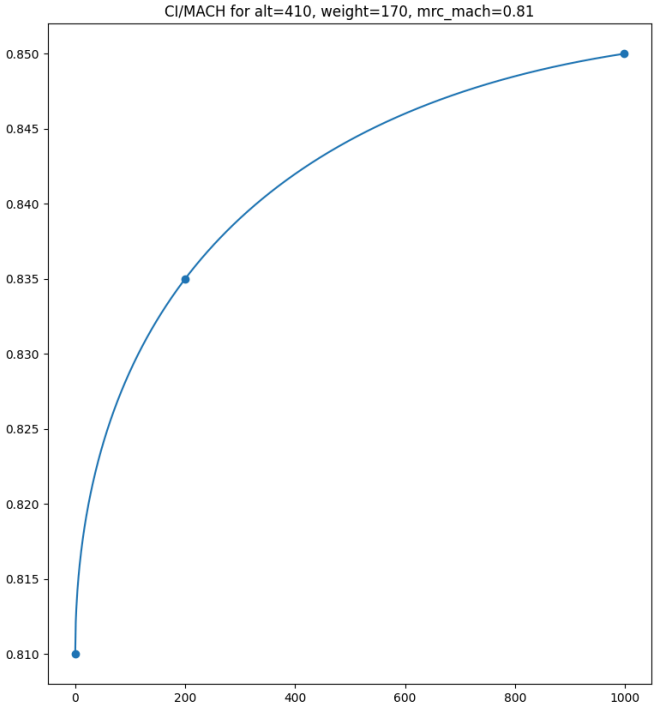




Sanity check: 0.804 MRC and 0.833 LRC

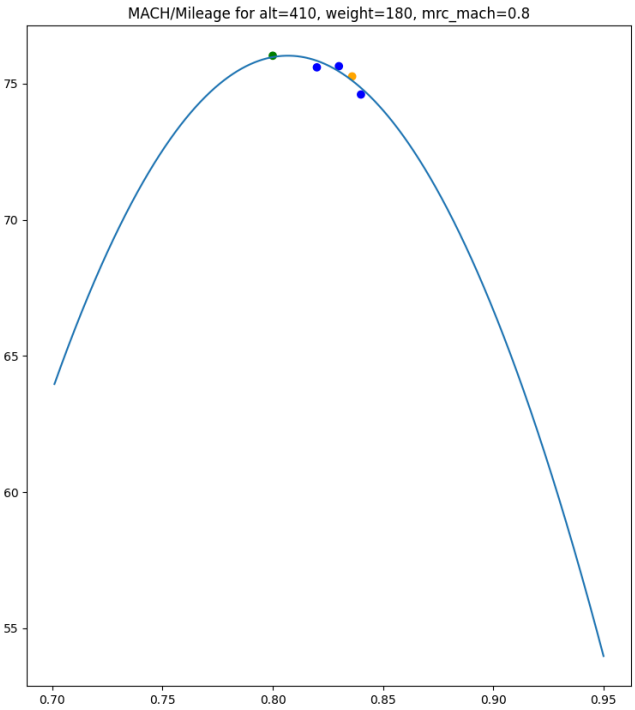
Found: 0.81 MRC and 0.835 LRC

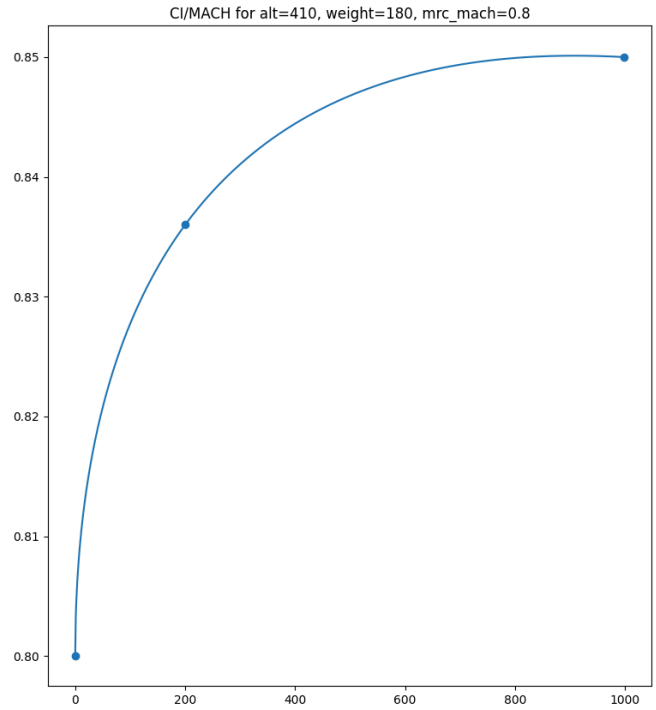




Sanity check: 0.81 MRC and 0.835 LRC

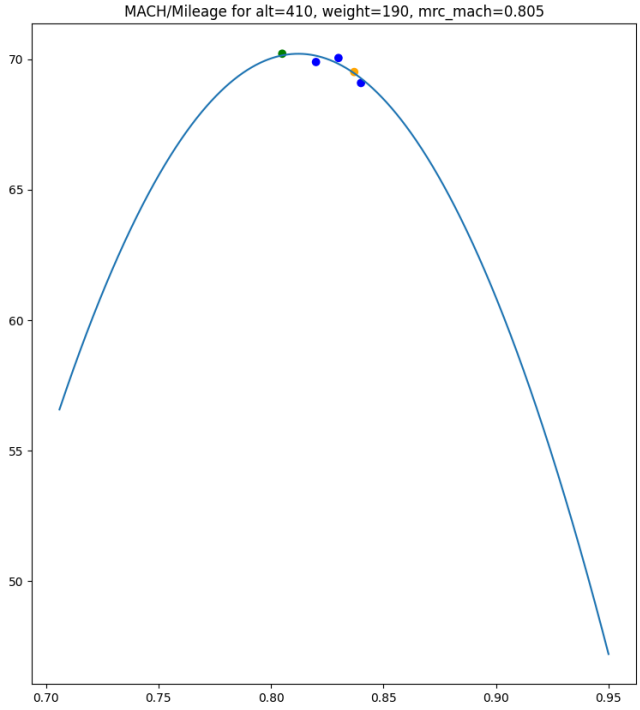
Found: 0.8 MRC and 0.836 LRC

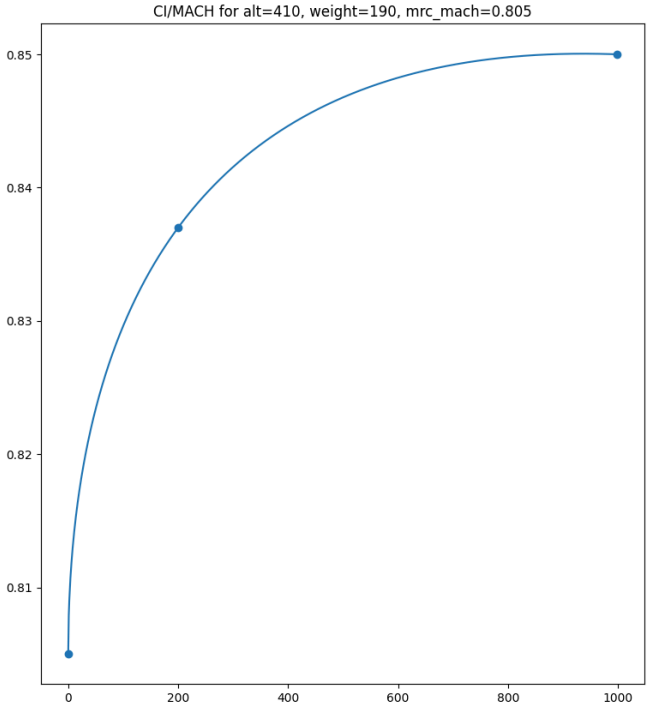




Sanity check: 0.8 MRC and 0.836 LRC

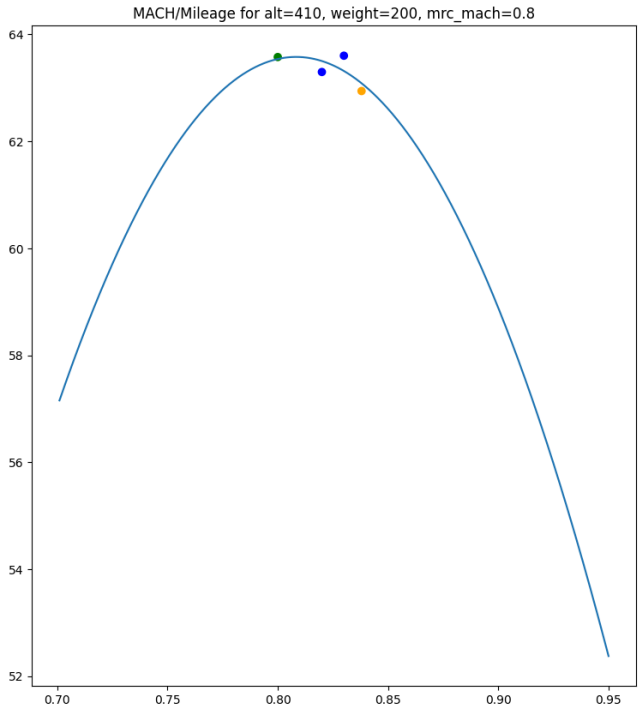
Found: 0.805 MRC and 0.837 LRC

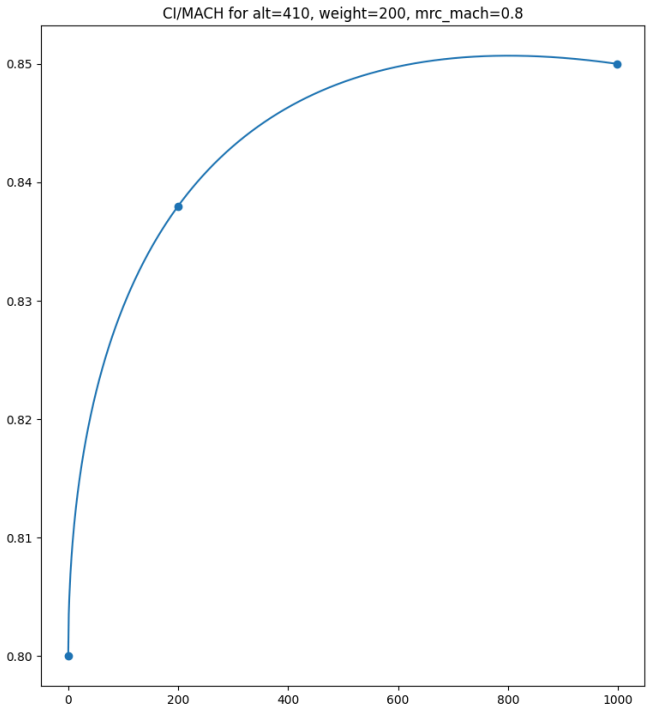




Sanity check: 0.805 MRC and 0.837 LRC

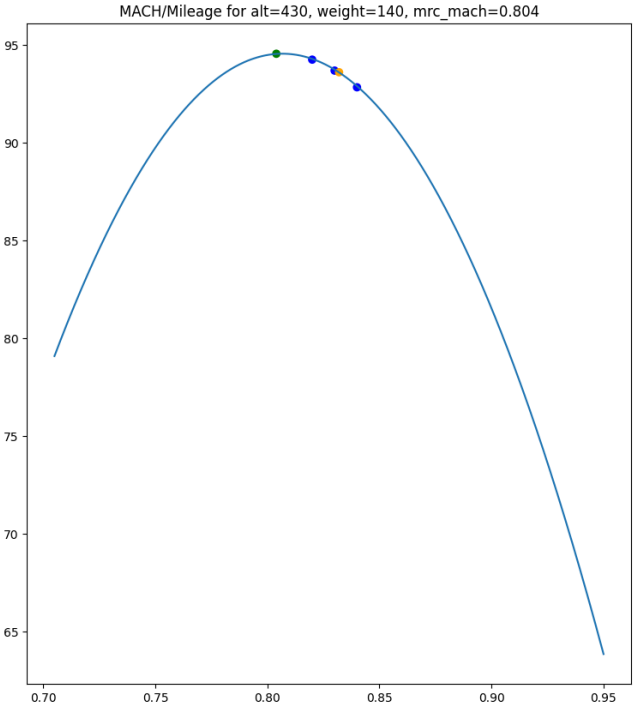
Found: 0.8 MRC and 0.838 LRC

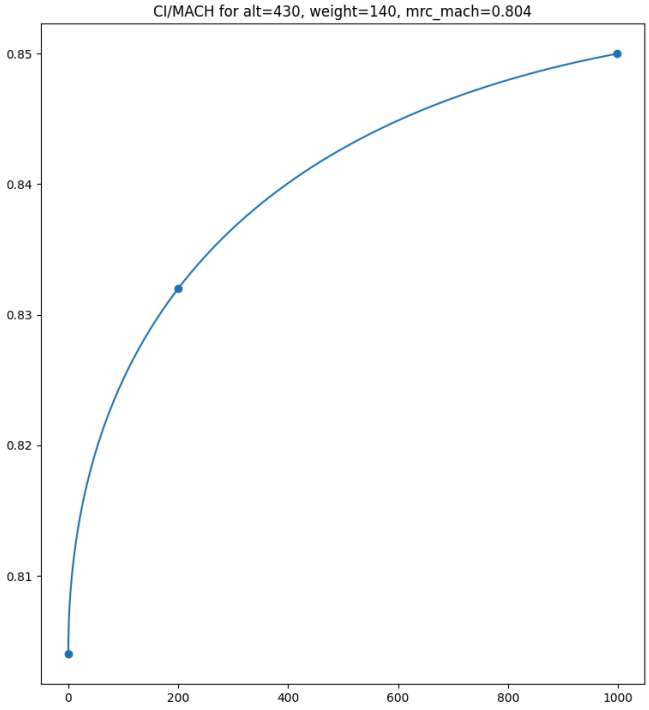




Sanity check: 0.8 MRC and 0.838 LRC

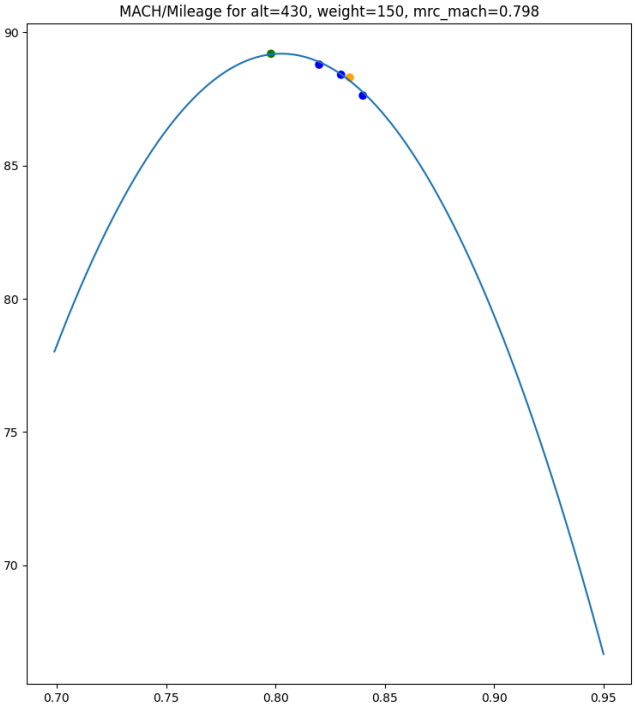
Found: 0.804 MRC and 0.832 LRC

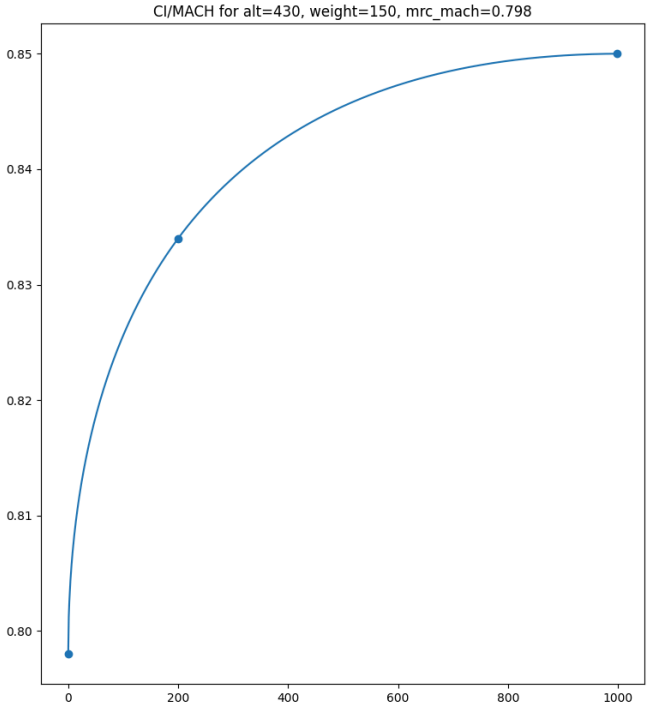




Sanity check: 0.804 MRC and 0.832 LRC

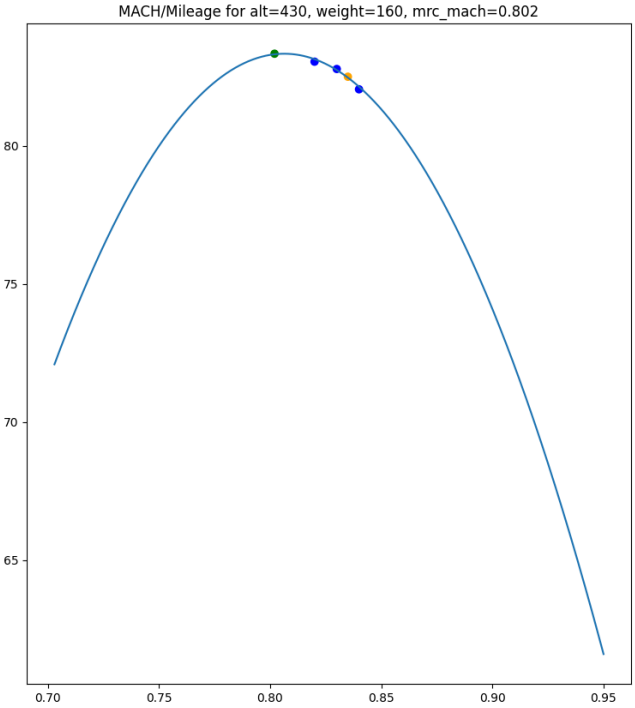
Found: 0.798 MRC and 0.834 LRC

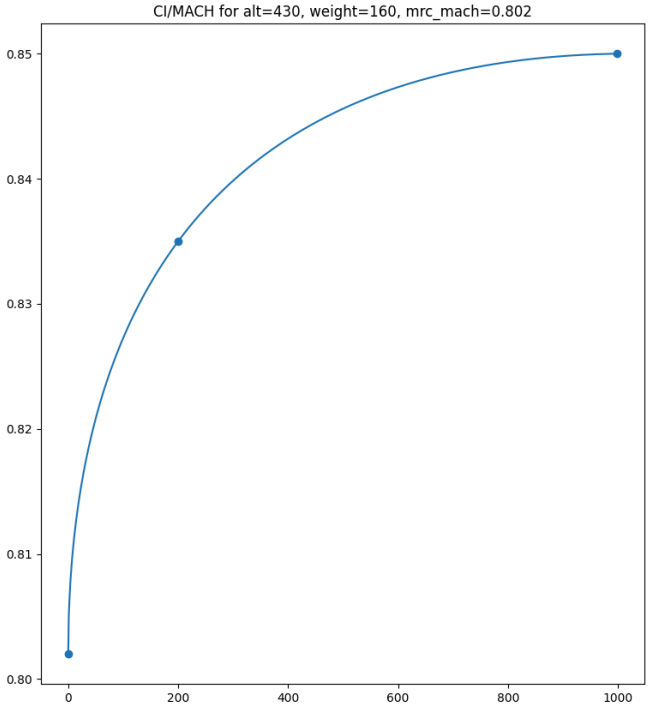




Sanity check: 0.798 MRC and 0.834 LRC

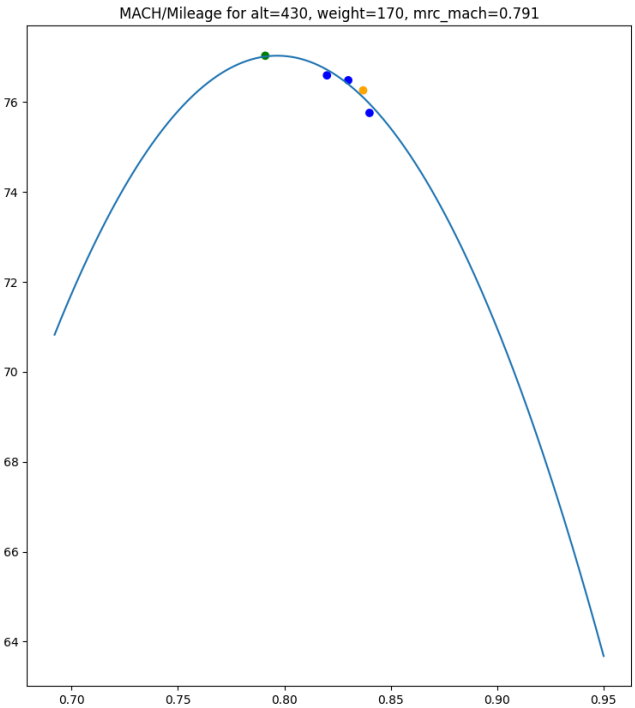
Found: 0.802 MRC and 0.835 LRC

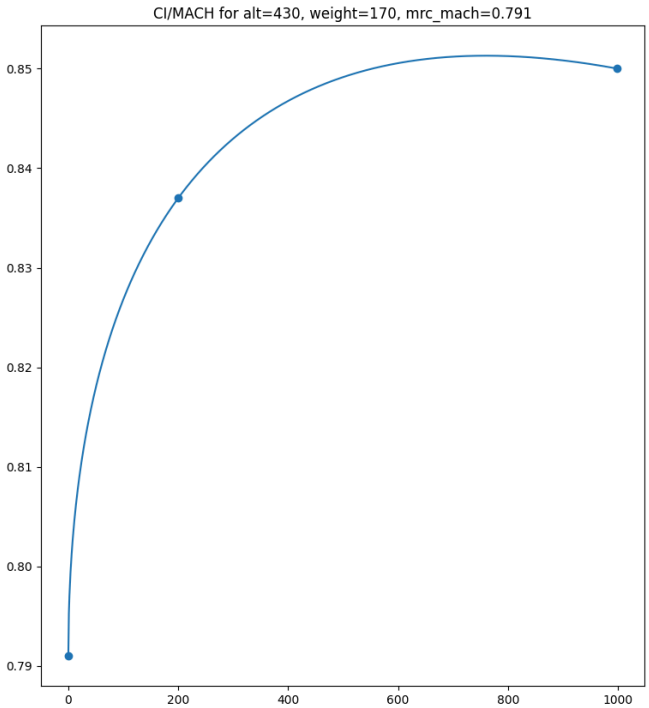




Sanity check: 0.802 MRC and 0.835 LRC

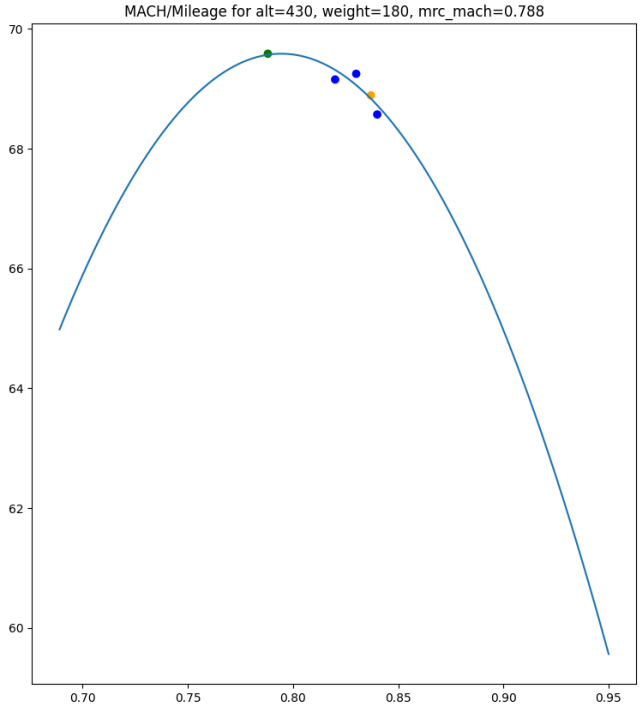
Found: 0.791 MRC and 0.837 LRC

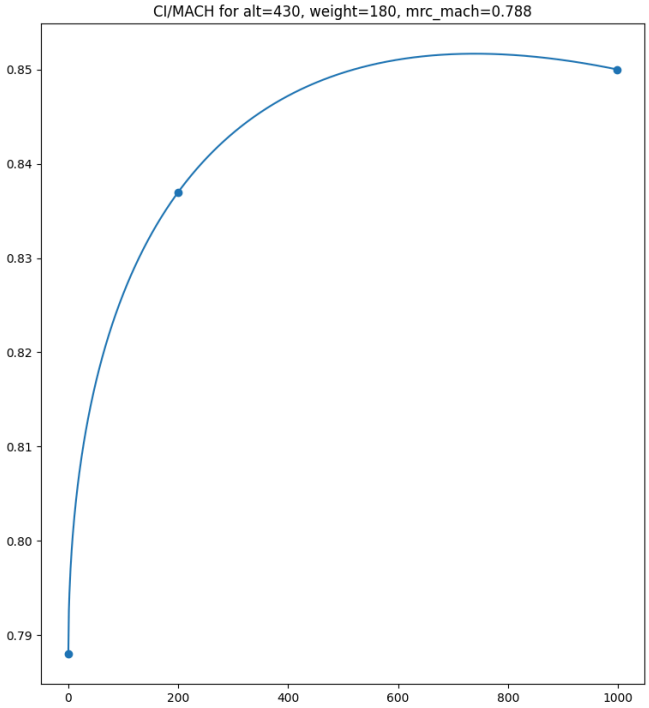




Sanity check: 0.791 MRC and 0.837 LRC

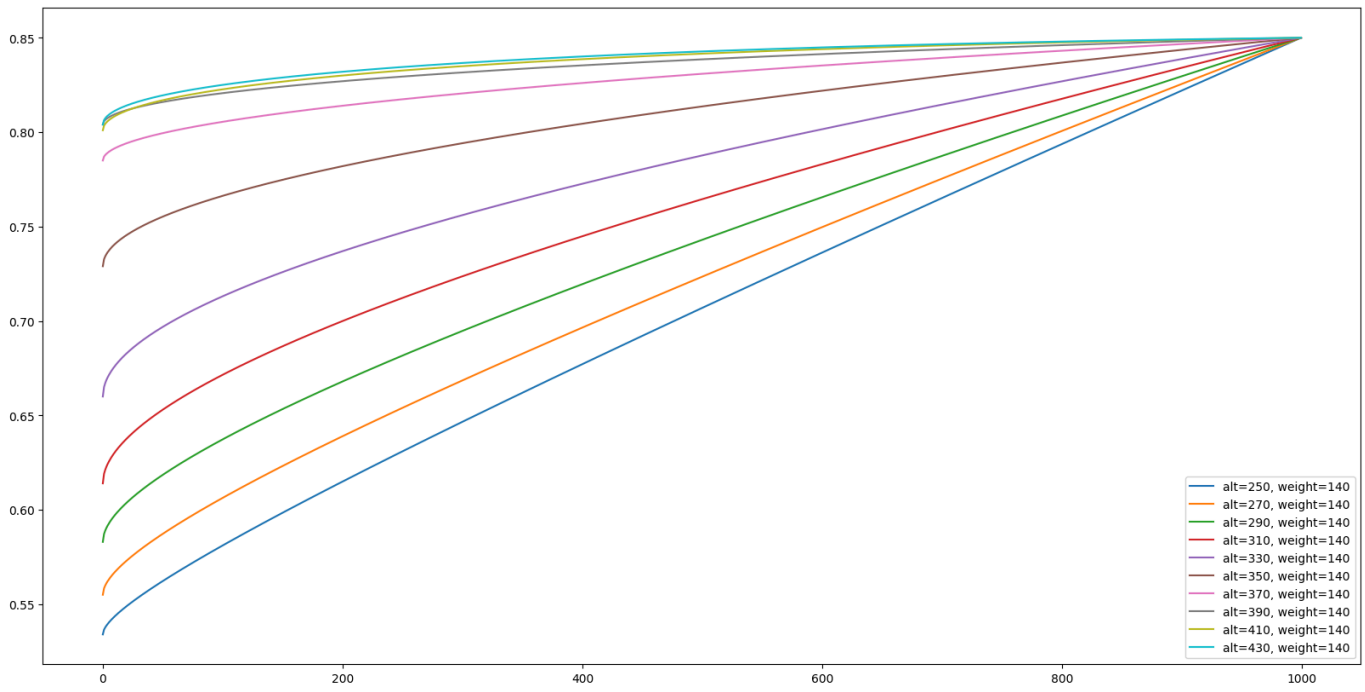
Found: 0.788 MRC and 0.837 LRC

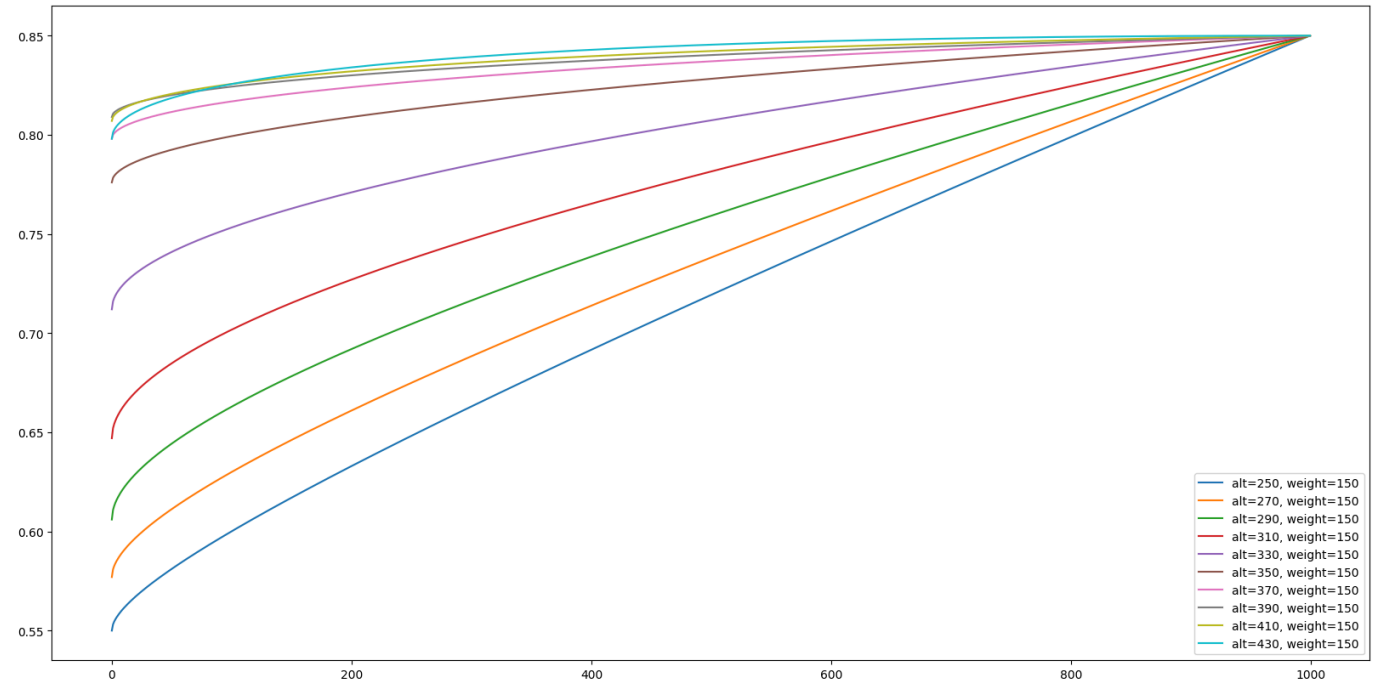


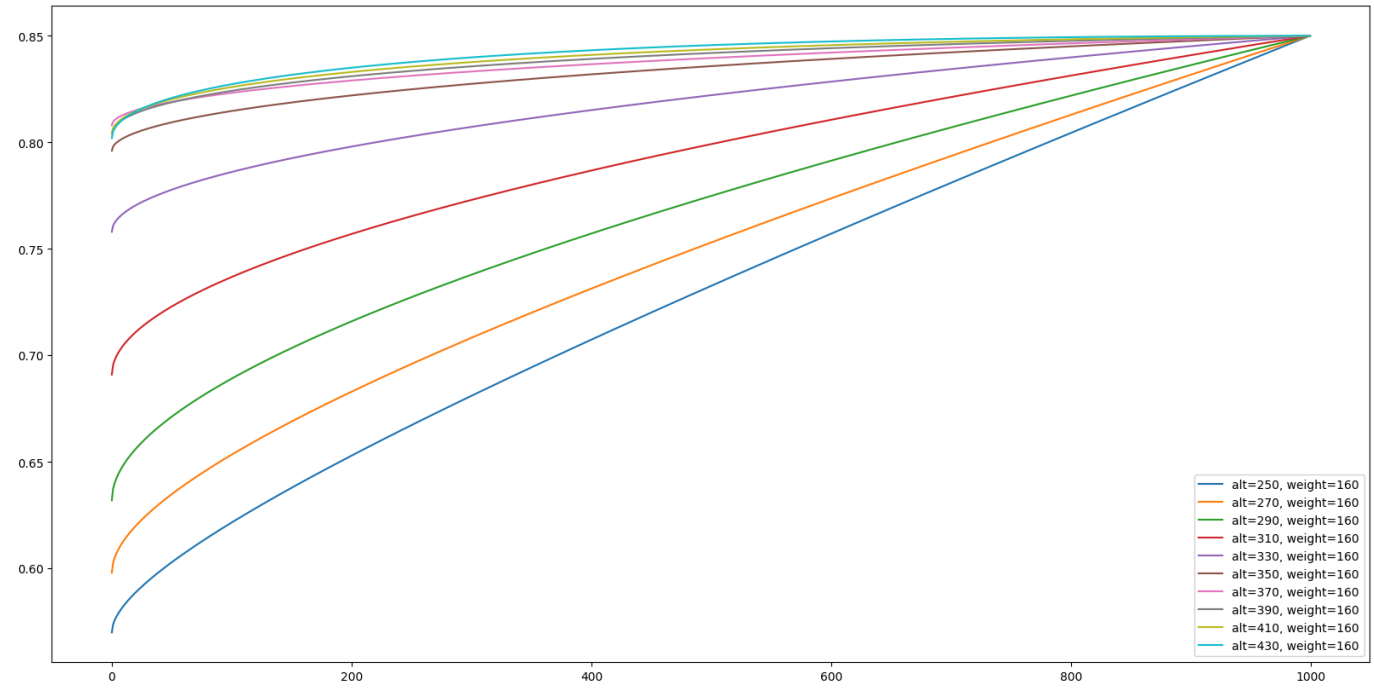


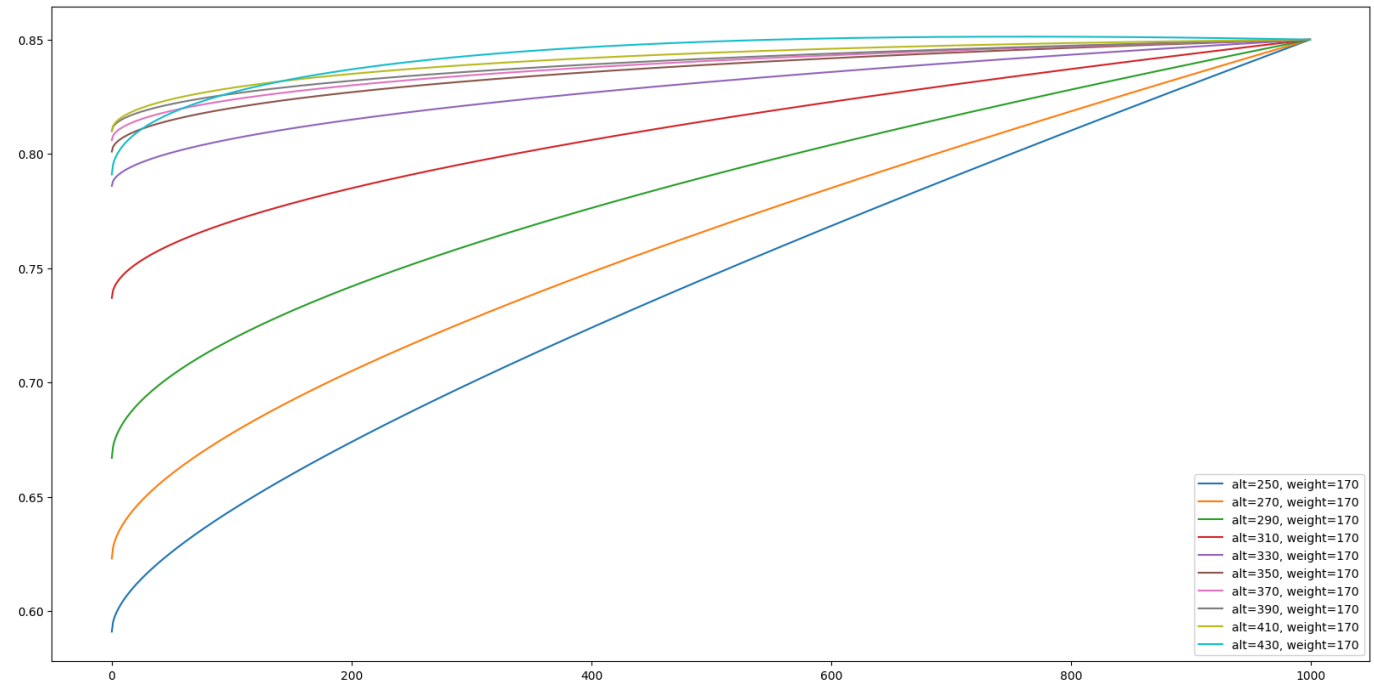
Sanity check: 0.788 MRC and 0.837 LRC

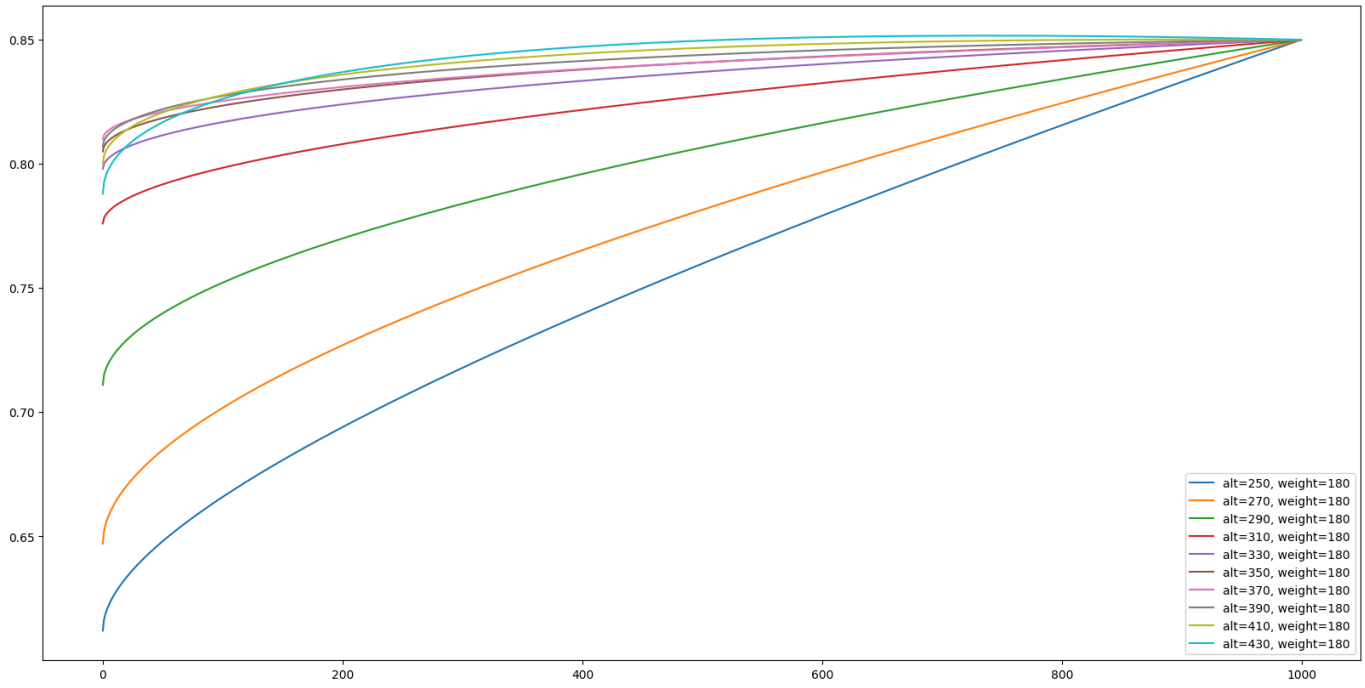
In [ ]:
```
if not skip_plots:
  x = np.linspace(0, 999, points_for_fit)

  for weight in  range(16):
    plt.figure(figsize=(20,10))
    for alt in range(0, 10):
      try:
        plt.plot(x, curves[alt][weight], label=f"alt={250 + alt * 20}, weight={140 + weight *
      except IndexError:
        pass
    plt.legend()
    plt.show()
```

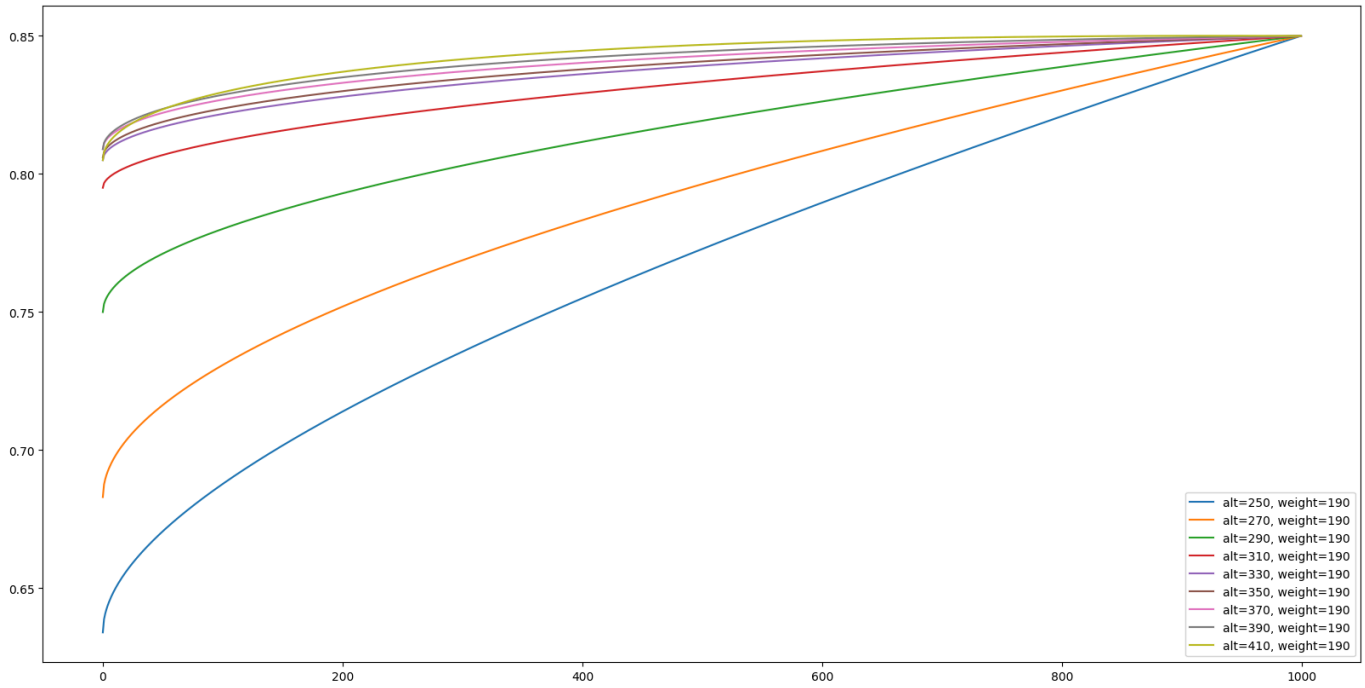


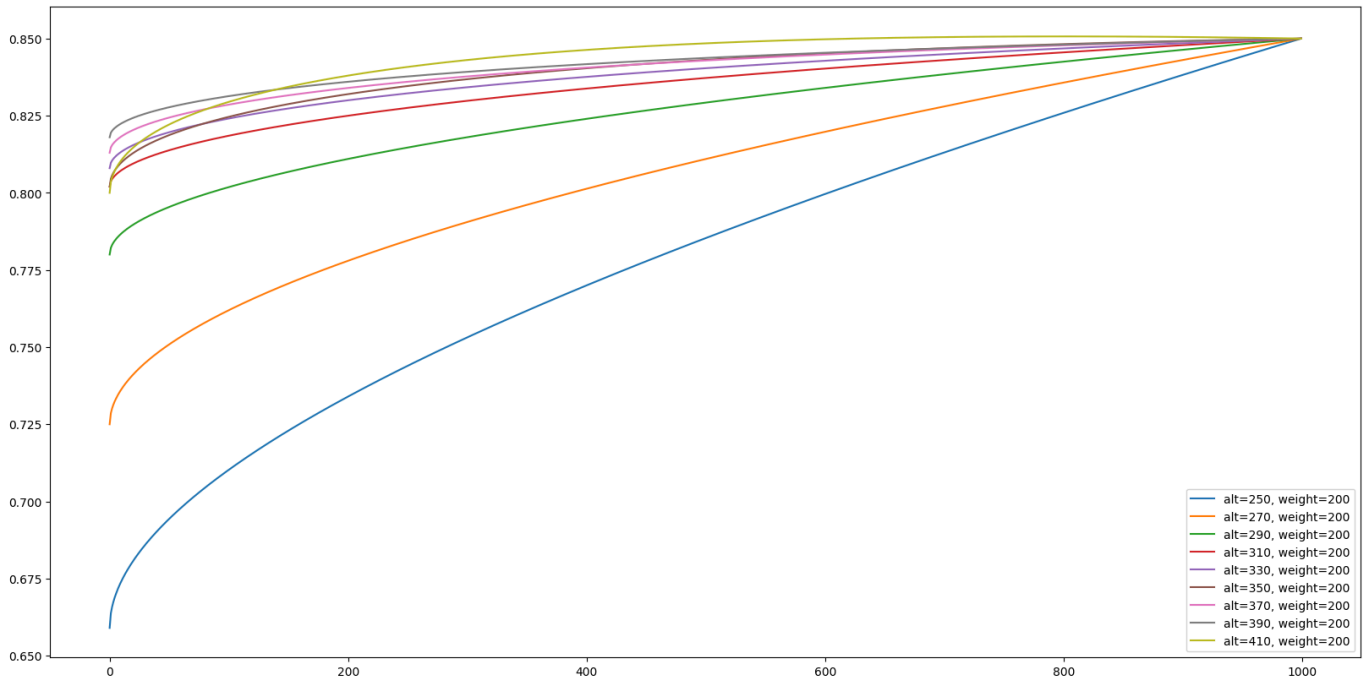

0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
200
400
600
800
1000
alt=250, weight=150
alt=270, weight=150
alt=290, weight=150
alt=310, weight=150
alt=330, weight=150
alt=350, weight=150
alt=370, weight=150
alt=390, weight=150
alt=410, weight=150
alt=430, weight=150

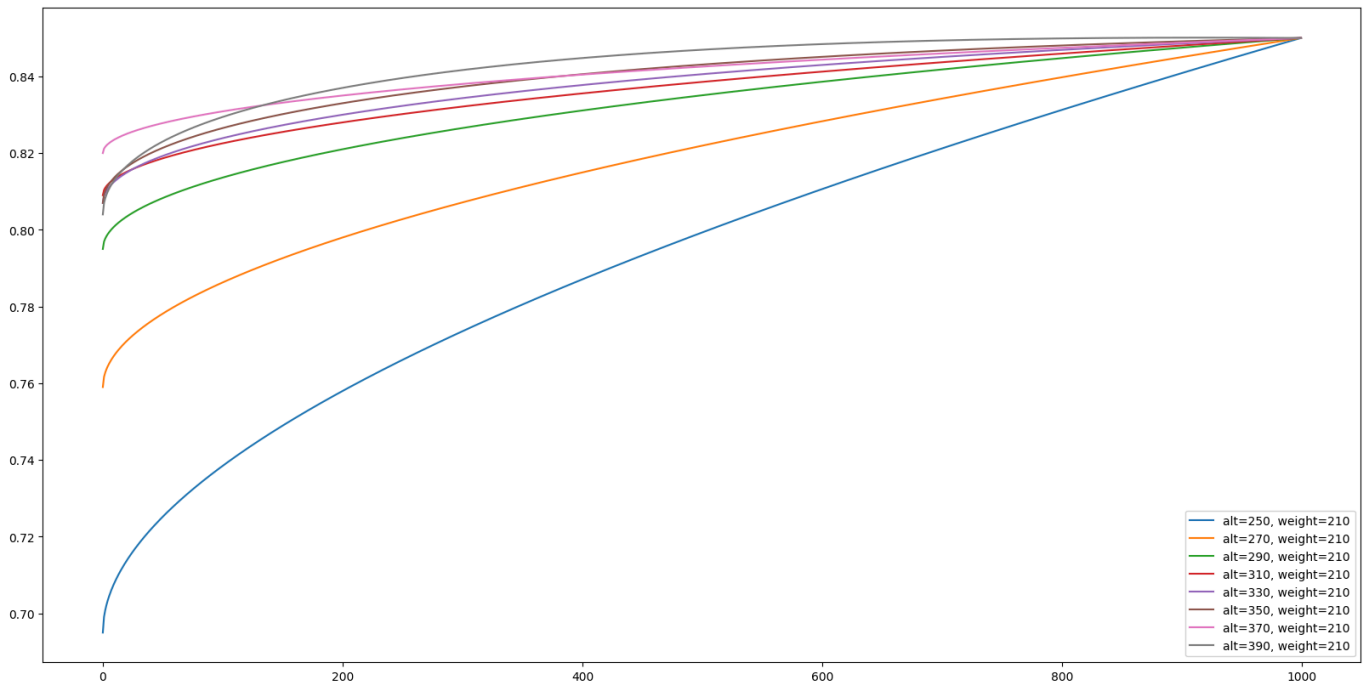

0.85
0.80
0.75
0.70
0.65
0.60
0
200
400
600
800
1000
alt=250, weight=160
alt=270, weight=160
alt=290, weight=160
alt=310, weight=160
alt=330, weight=160
alt=350, weight=160
alt=370, weight=160
alt=390, weight=160
alt=410, weight=160
alt=430, weight=160

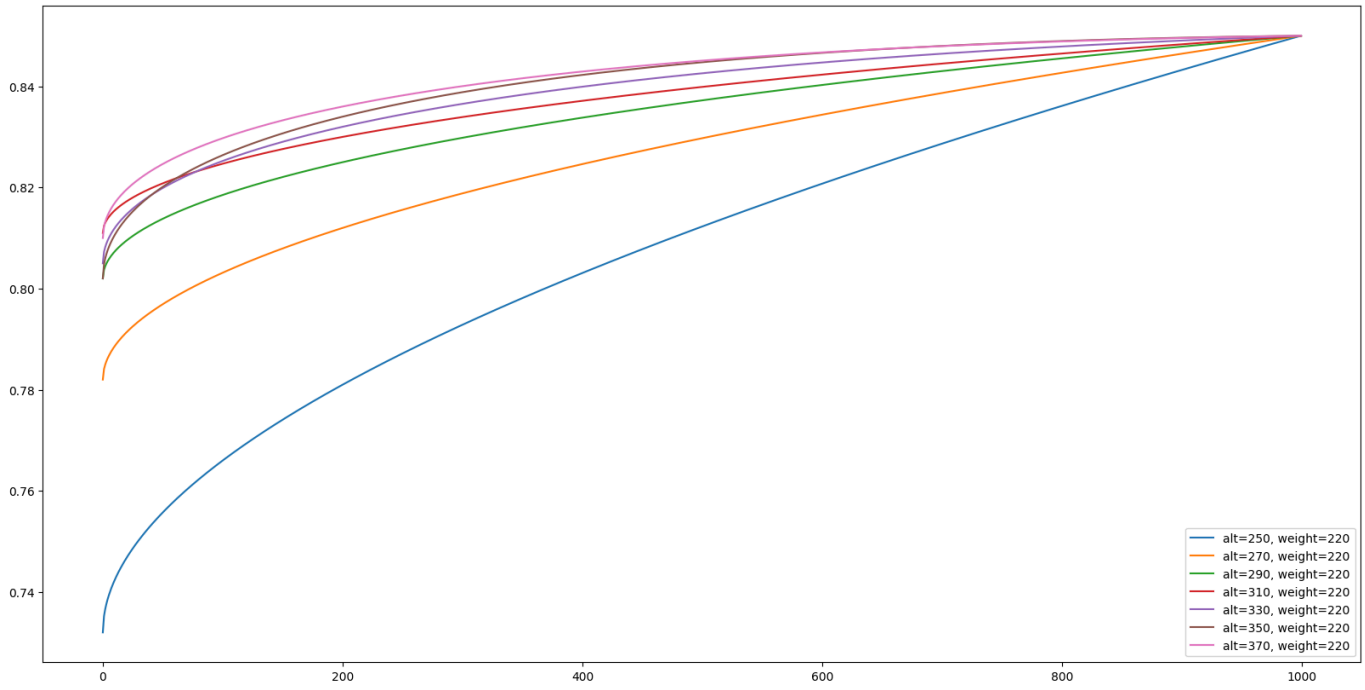

0.85
0.80
0.75
0.70
0.65
0.60
0
200
400
600
800
1000
alt=250, weight=170
alt=270, weight=170
alt=290, weight=170
alt=310, weight=170
alt=330, weight=170
alt=350, weight=170
alt=370, weight=170
alt=390, weight=170
alt=410, weight=170
alt=430, weight=170

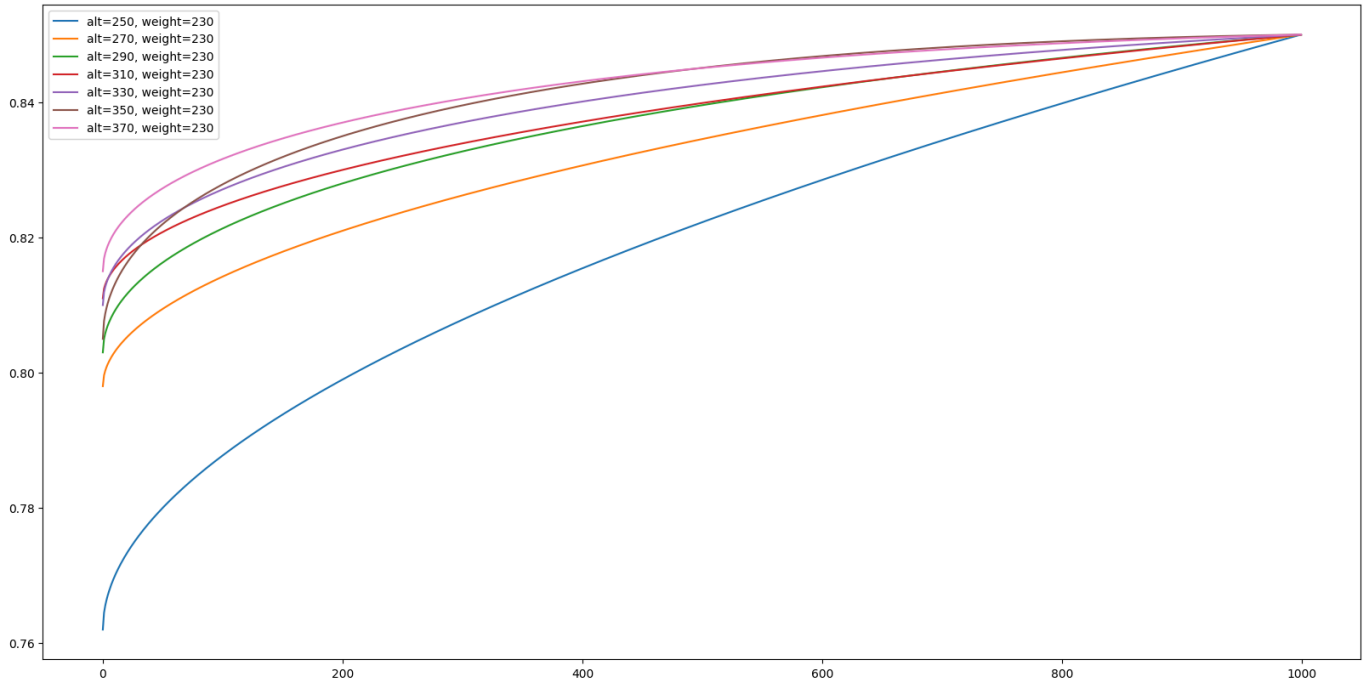

0.85
0.80
0.75
0.70
0.65
0
200
400
600
800
1000
alt=250, weight=180
alt=270, weight=180
alt=290, weight=180
alt=310, weight=180
alt=330, weight=180
alt=350, weight=180
alt=370, weight=180
alt=390, weight=180
alt=410, weight=180
alt=430, weight=180

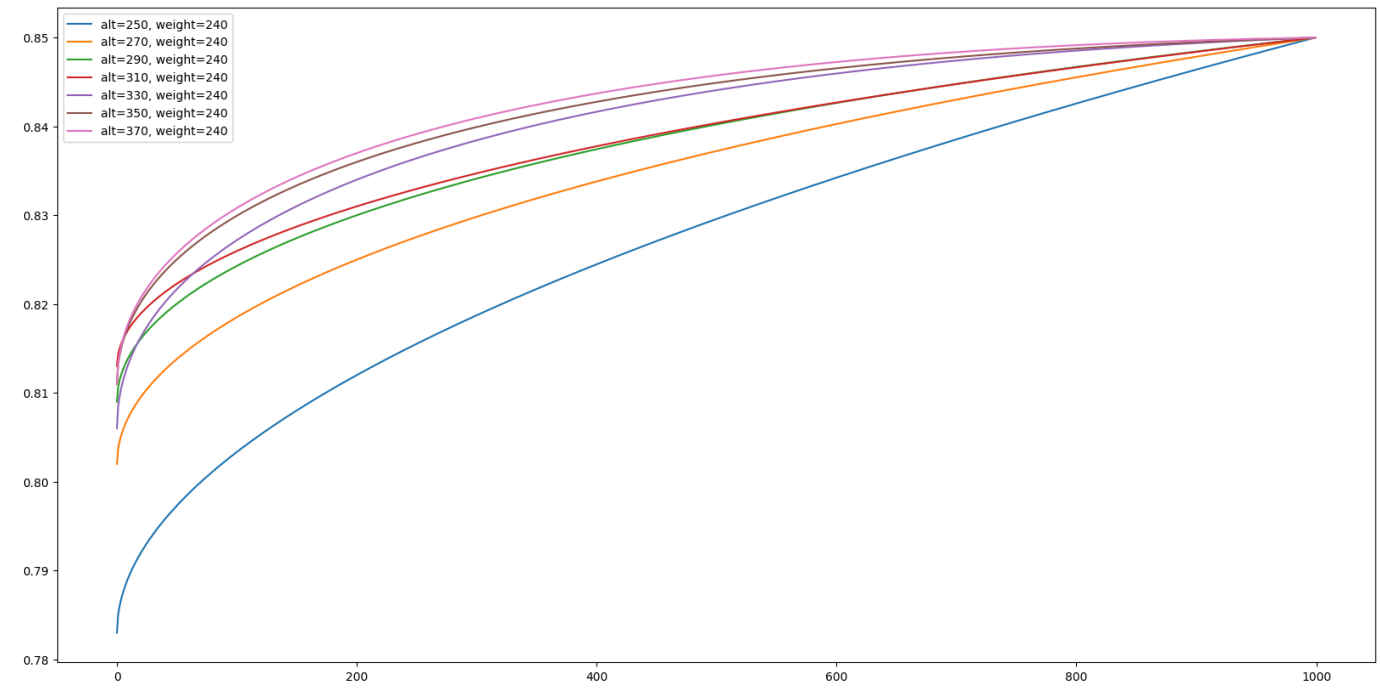

0.85
0.80
0.75
0.70
0.65
0
200
400
600
800
1000
alt=250, weight=190
alt=270, weight=190
alt=290, weight=190
alt=310, weight=190
alt=330, weight=190
alt=350, weight=190
alt=370, weight=190
alt=390, weight=190
alt=410, weight=190

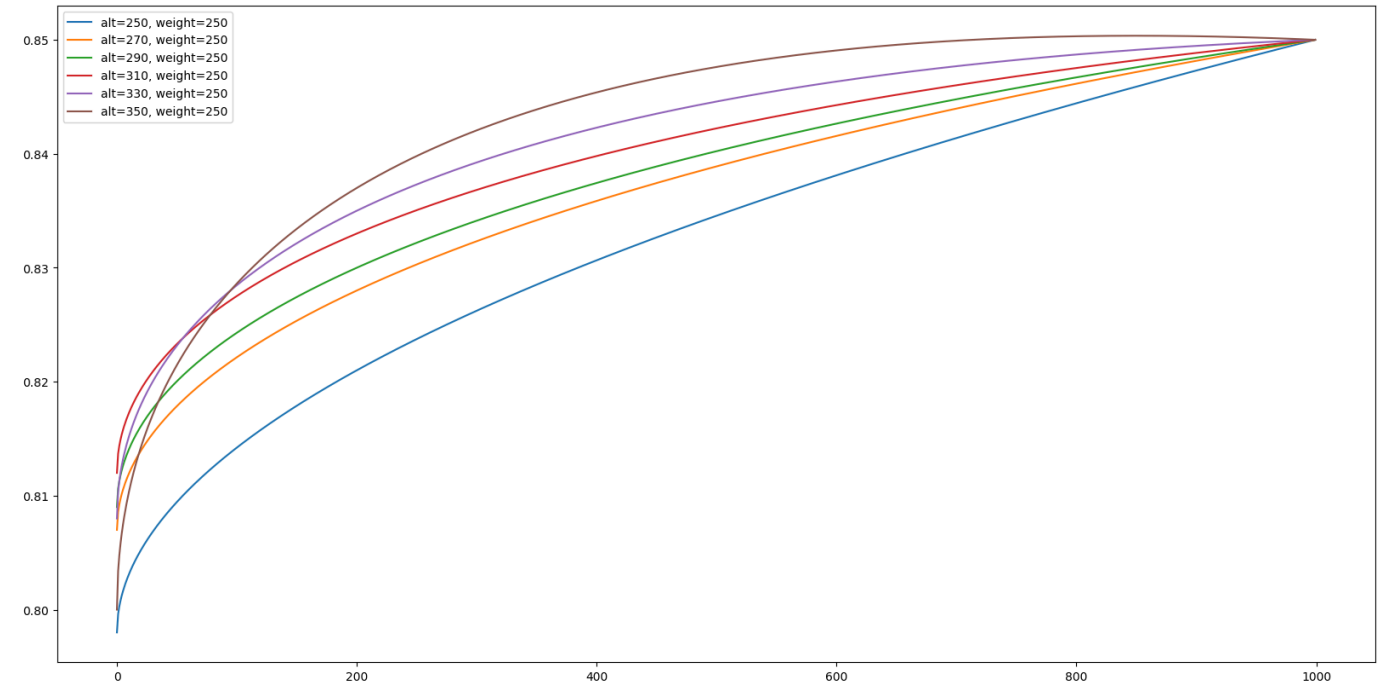

0.850
0.825
0.800
0.775
0.750
0.725
0.700
0.675
0.650
0
200
400
600
800
1000
alt=250, weight=200
alt=270, weight=200
alt=290, weight=200
alt=310, weight=200
alt=330, weight=200
alt=350, weight=200
alt=370, weight=200
alt=390, weight=200
alt=410, weight=200

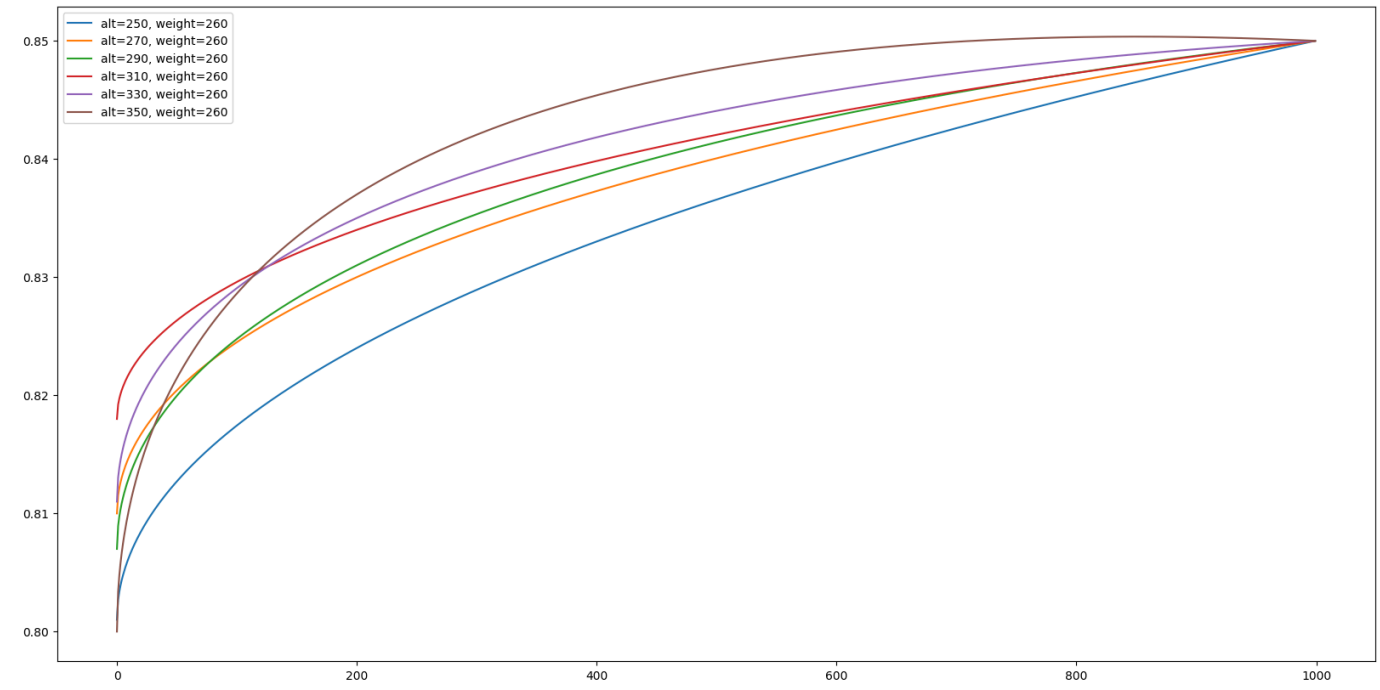

0.84
0.82
0.80
0.78
0.76
0.74
0.72
0.70
0
200
400
600
800
1000
alt=250, weight=210
alt=270, weight=210
alt=290, weight=210
alt=310, weight=210
alt=330, weight=210
alt=350, weight=210
alt=370, weight=210
alt=390, weight=210

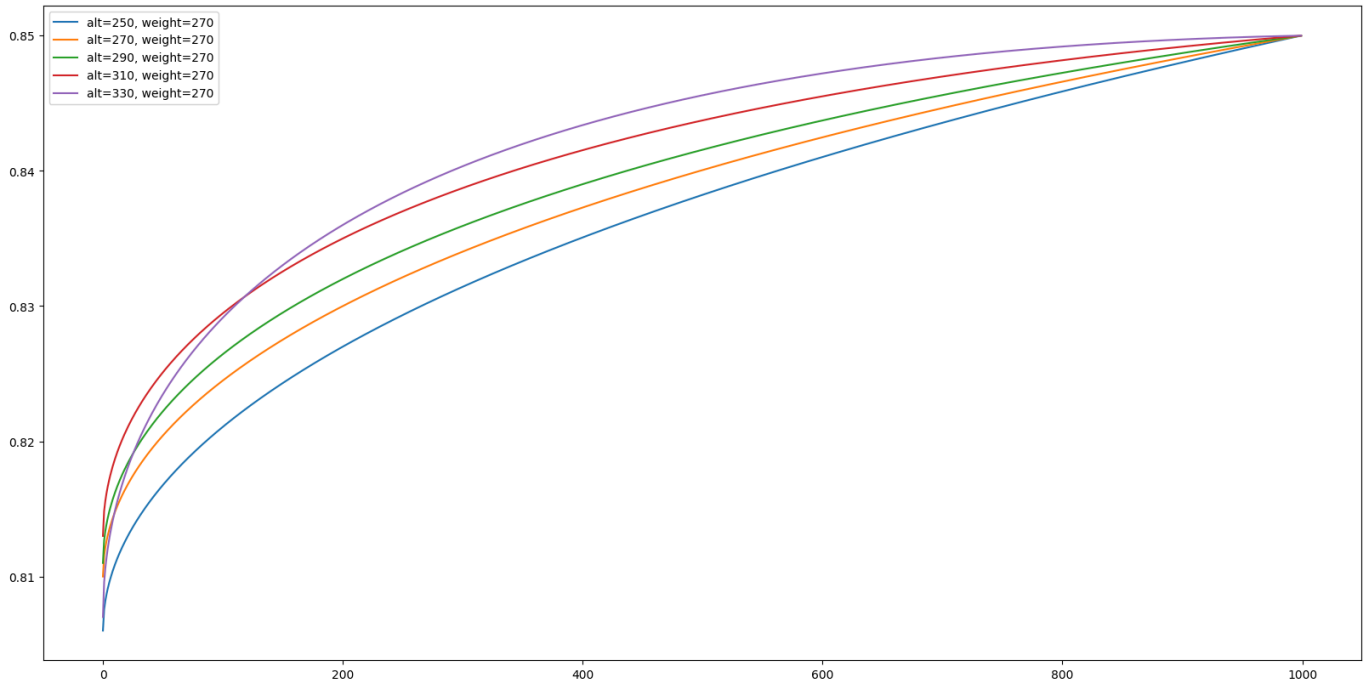

0.84
0.82
0.80
0.78
0.76
0.74
0
200
400
600
800
1000
alt=250, weight=220
alt=270, weight=220
alt=290, weight=220
alt=310, weight=220
alt=330, weight=220
alt=350, weight=220
alt=370, weight=220

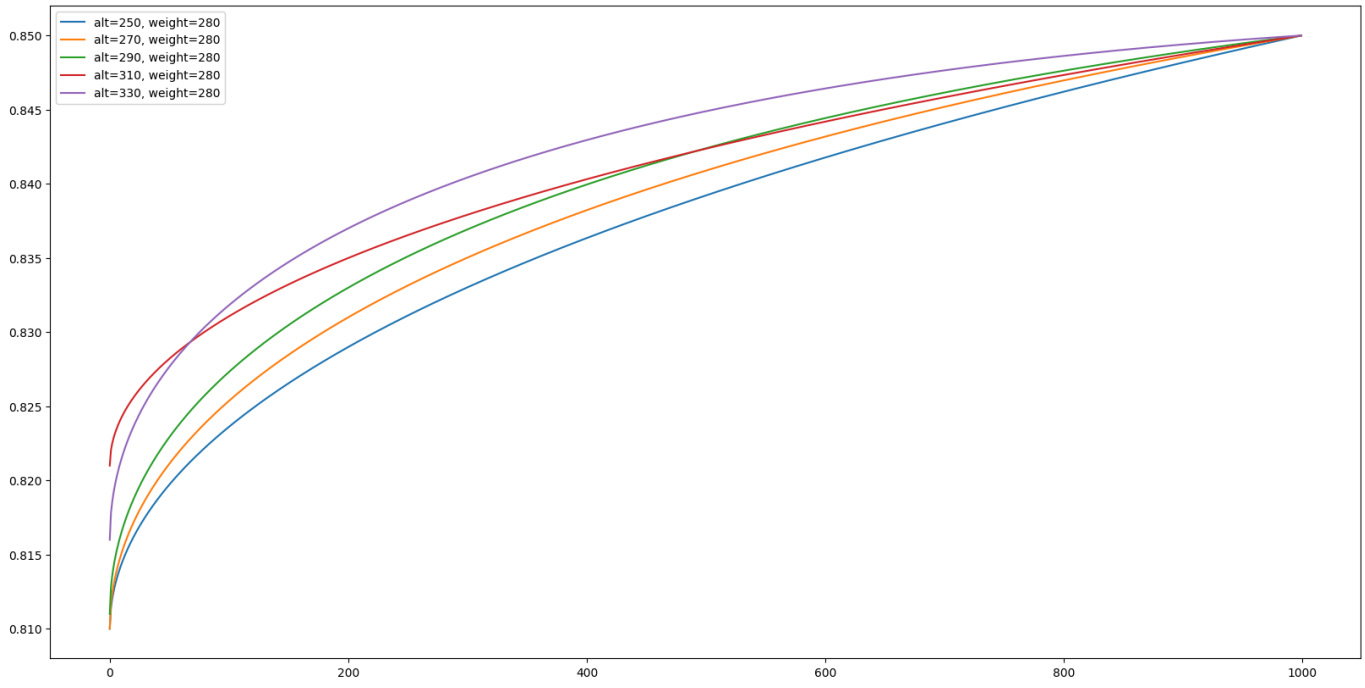

alt=250, weight=230
alt=270, weight=230
alt=290, weight=230
alt=310, weight=230
alt=330, weight=230
alt=350, weight=230
alt=370, weight=230
0.84
0.82
0.80
0.78
0.76
0
200
400
600
800
1000

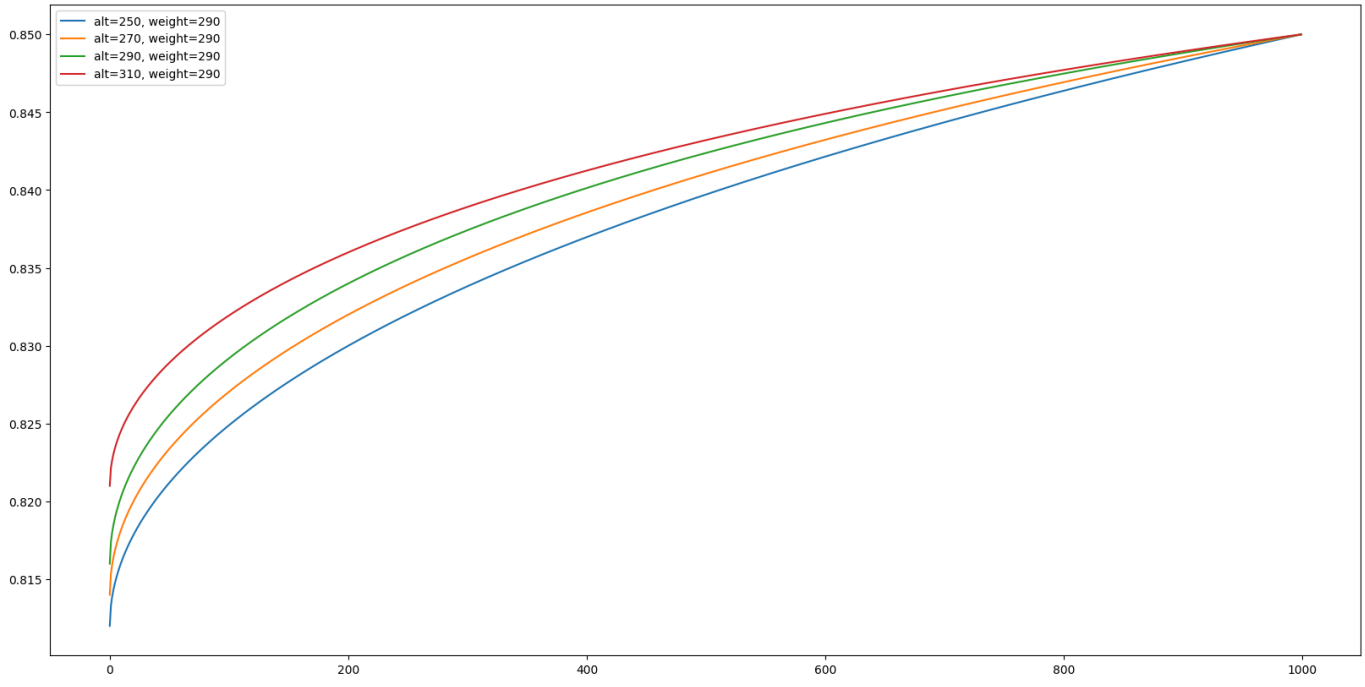

alt=250, weight=240
alt=270, weight=240
alt=290, weight=240
alt=310, weight=240
alt=330, weight=240
alt=350, weight=240
alt=370, weight=240
0.85
0.84
0.83
0.82
0.81
0.80
0.79
0.78
0
200
400
600
800
1000


alt=250, weight=250
alt=270, weight=250
alt=290, weight=250
alt=310, weight=250
alt=330, weight=250
alt=350, weight=250
0.85
0.84
0.83
0.82
0.81
0.80
0
200
400
600
800
1000


alt=250, weight=260
alt=270, weight=260
alt=290, weight=260
alt=310, weight=260
alt=330, weight=260
alt=350, weight=260
0.85
0.84
0.83
0.82
0.81
0.80
0
200
400
600
800
1000

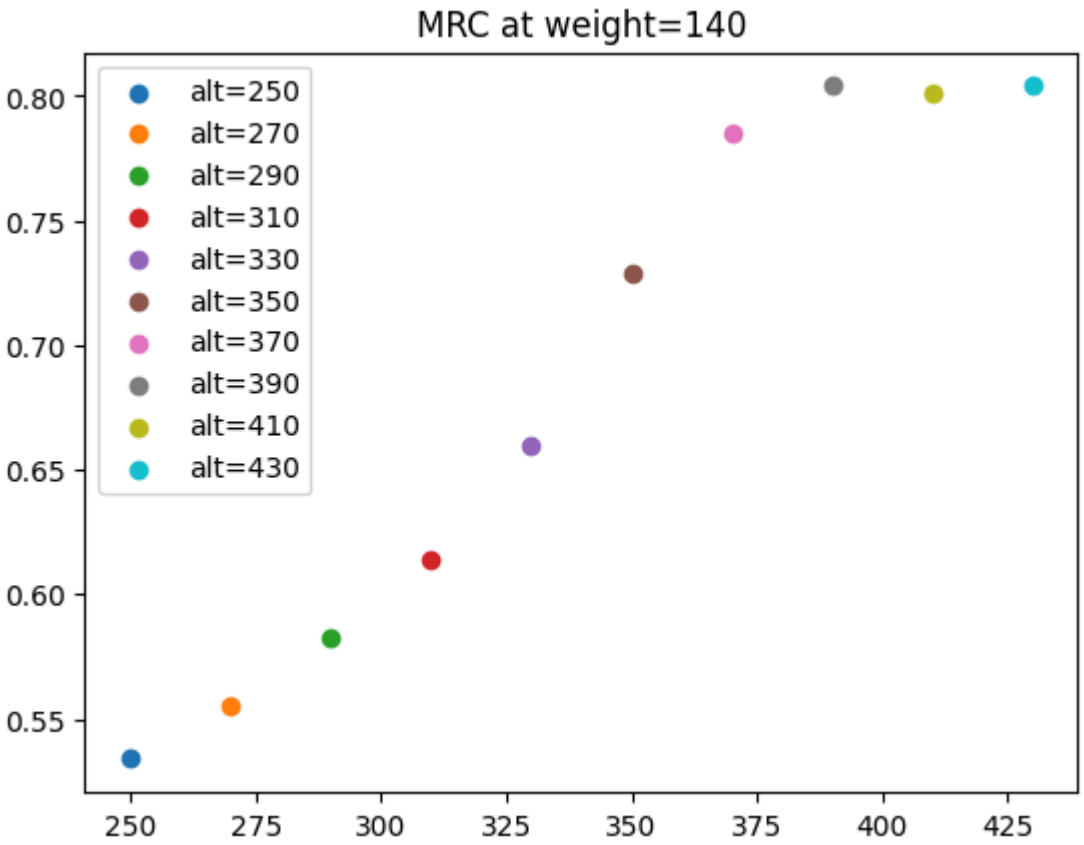


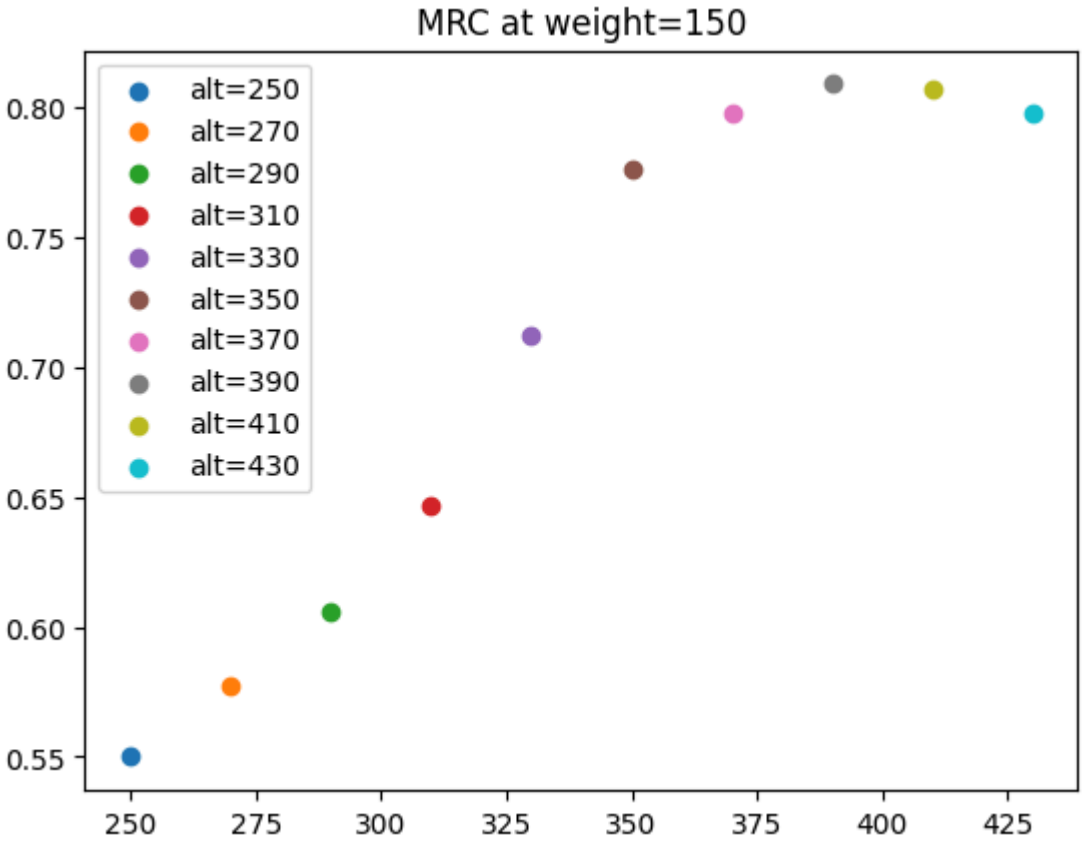


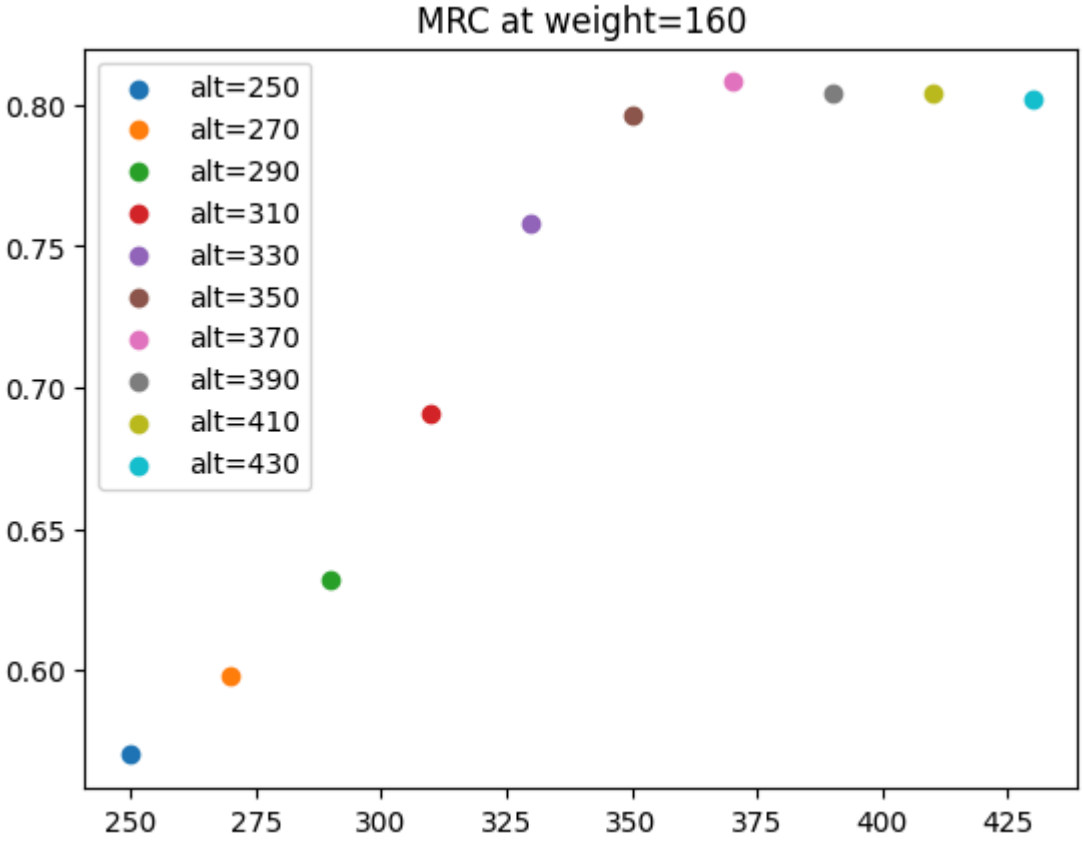


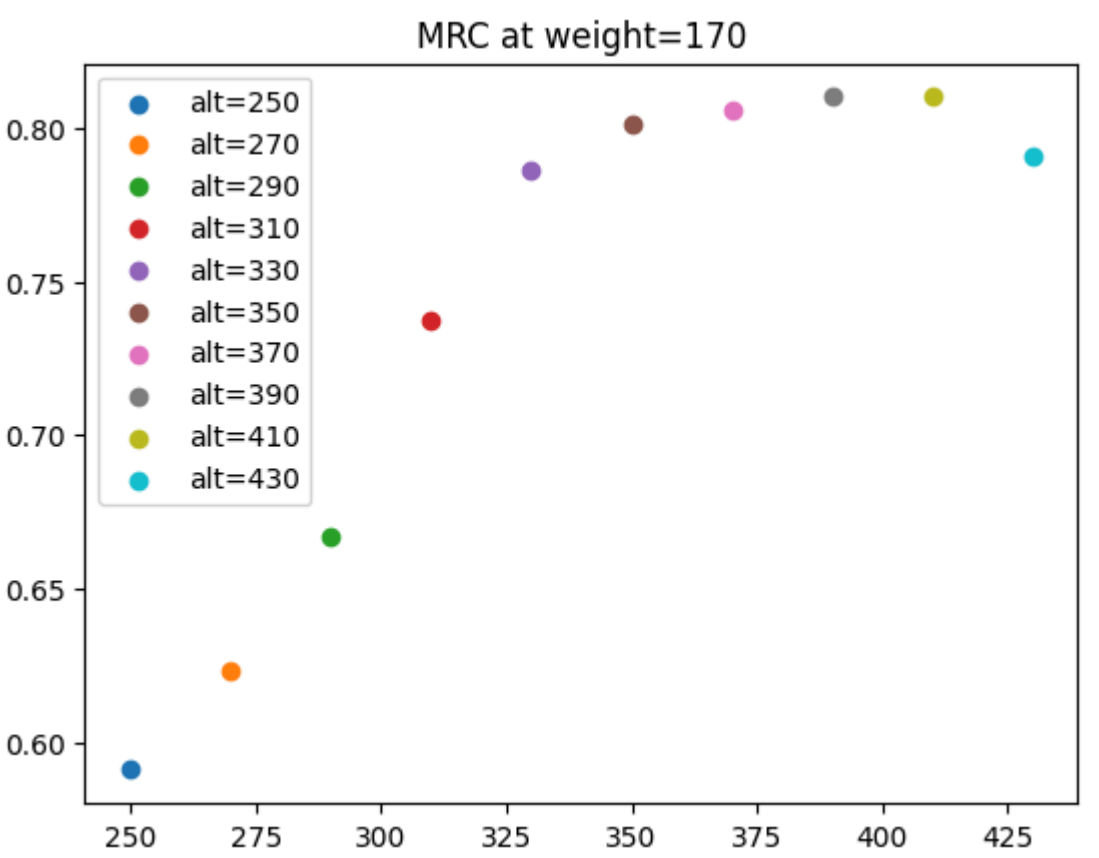
In [ ]:
```
# Indexing is altitude, weight
# (altitude - 250) / 20 to normalize
# (weight - 140) / 10 to normalize
```

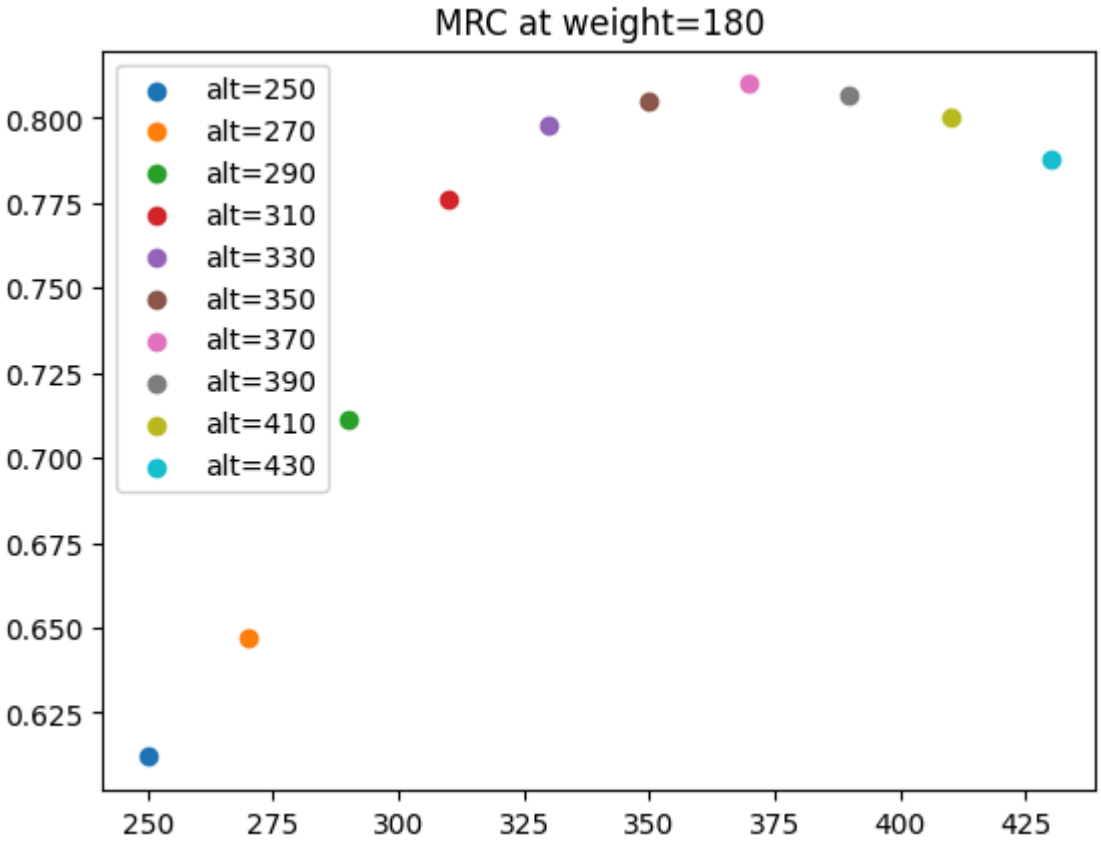
```
# altitude step in FL20
# weight step in 10t

if not skip_plots:
  x = np.linspace(250, 430, 10)

  for weight in range(16):
    plt.title(f"MRC at weight={140 + weight * 10}")
    for alt in range(10):
      try:
        plt.scatter(x[alt], curves[alt][weight][0], label=f"alt={250 + alt * 20}")
      except IndexError:
        pass
    plt.legend()
    plt.show()
```

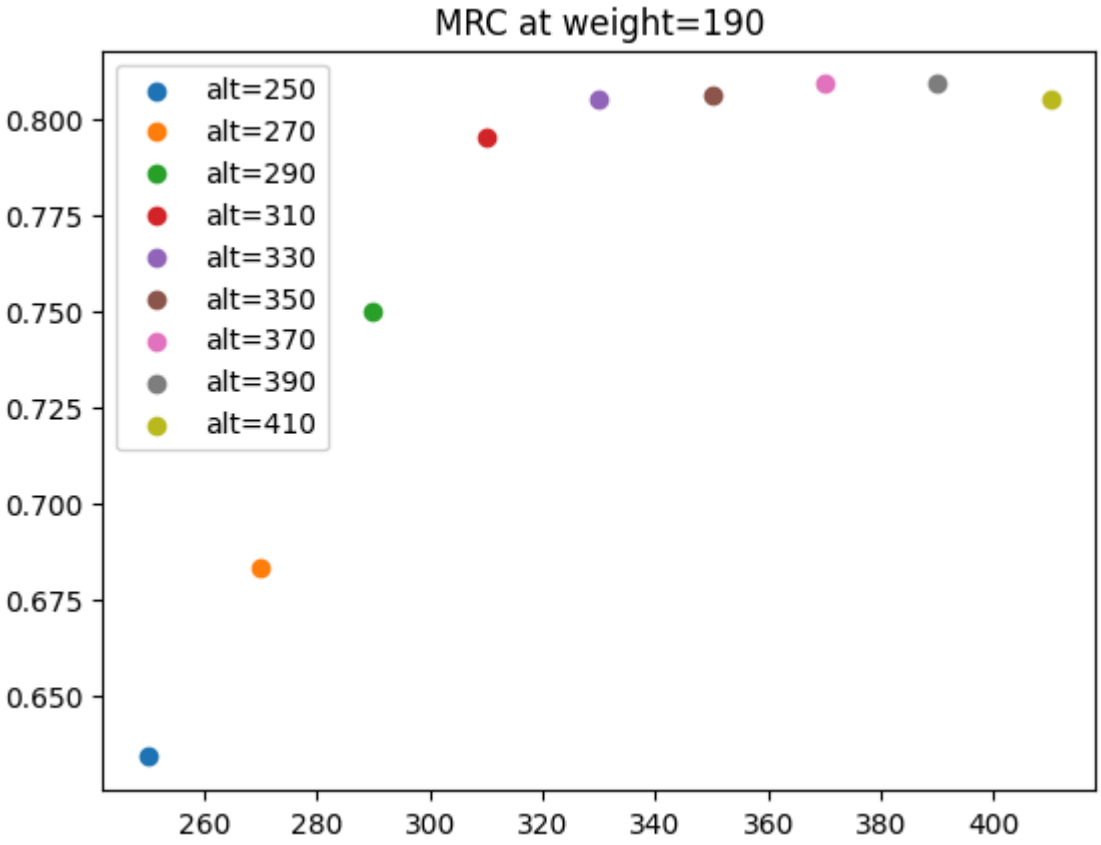


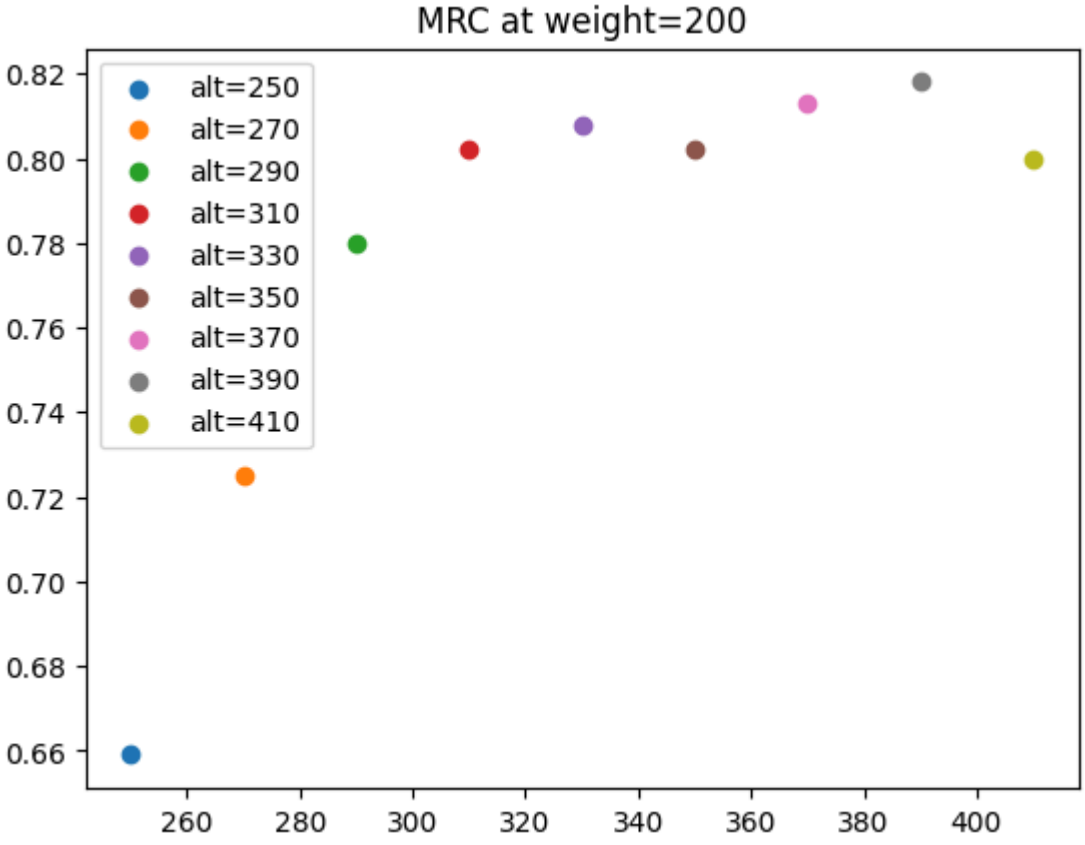

MRC at weight=150
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
alt=370
alt=390
alt=410
alt=430
0.80
0.75
0.70
0.65
0.60
0.55
250
275
300
325
350
375
400
425

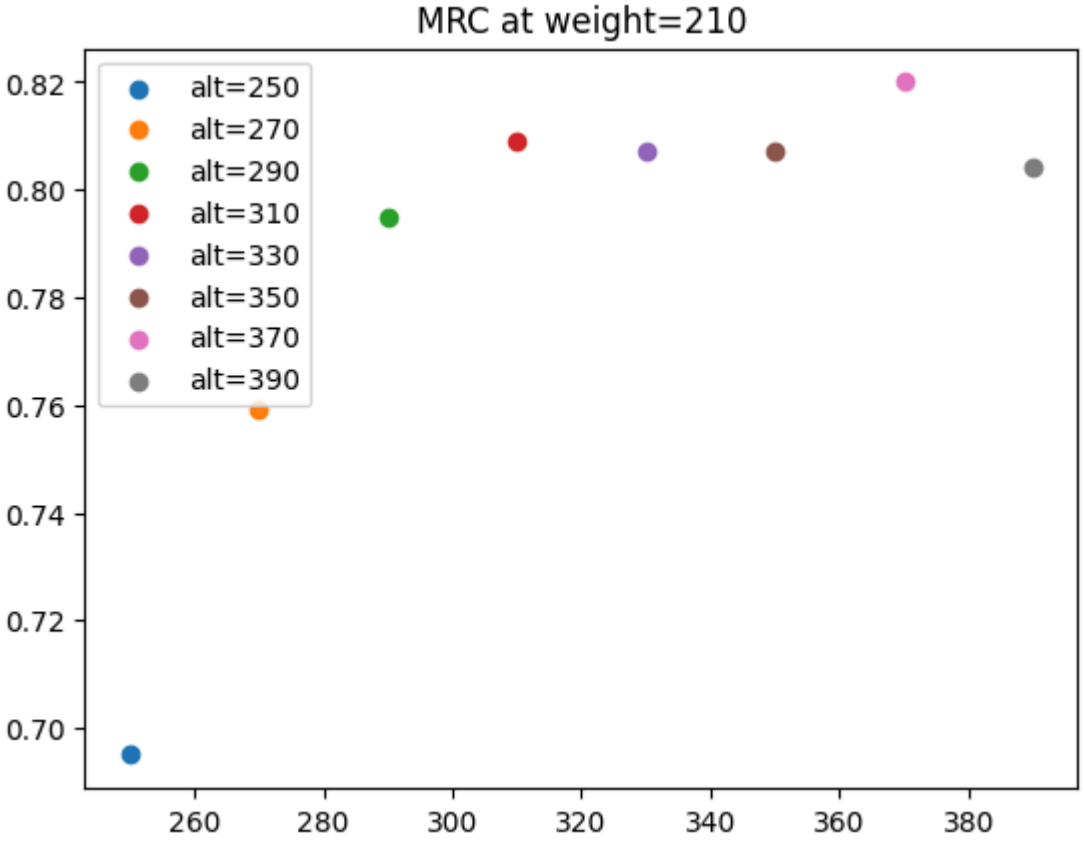

MRC at weight=160
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
alt=370
alt=390
alt=410
alt=430
0.80
0.75
0.70
0.65
0.60
250
275
300
325
350
375
400
425

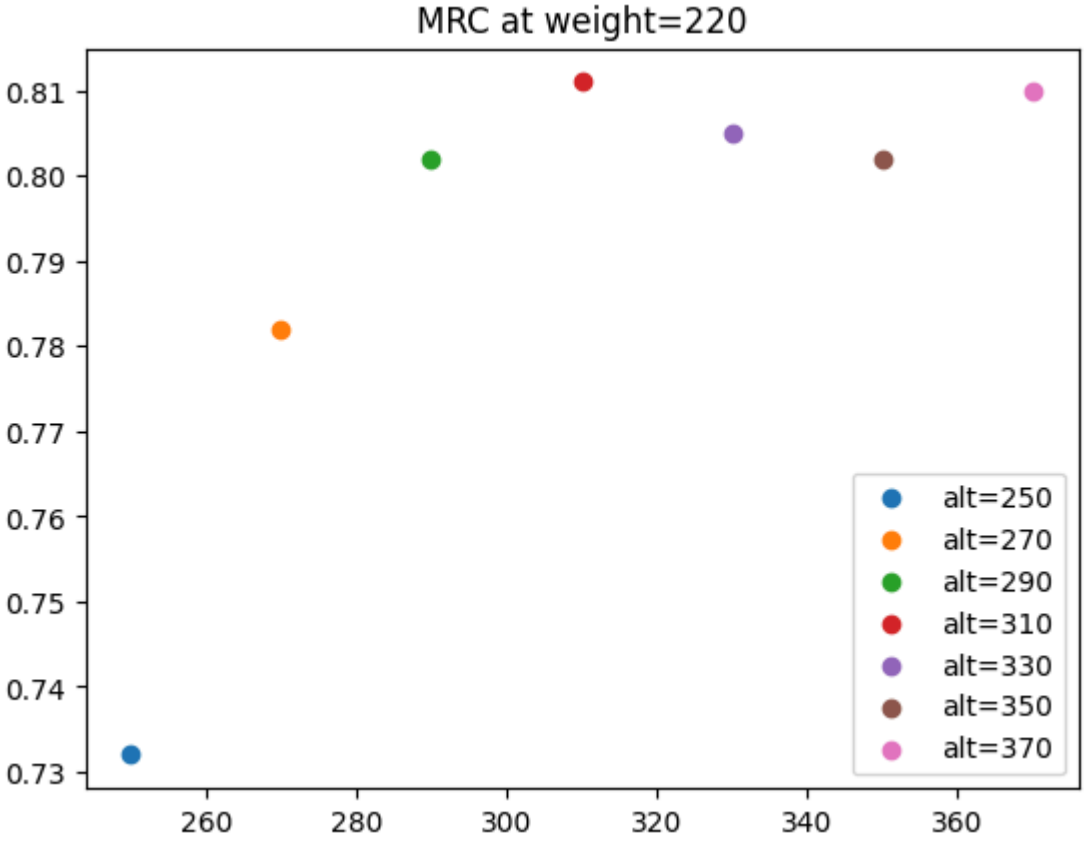

MRC at weight=170
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
alt=370
alt=390
alt=410
alt=430
0.80
0.75
0.70
0.65
0.60
250
275
300
325
350
375
400
425

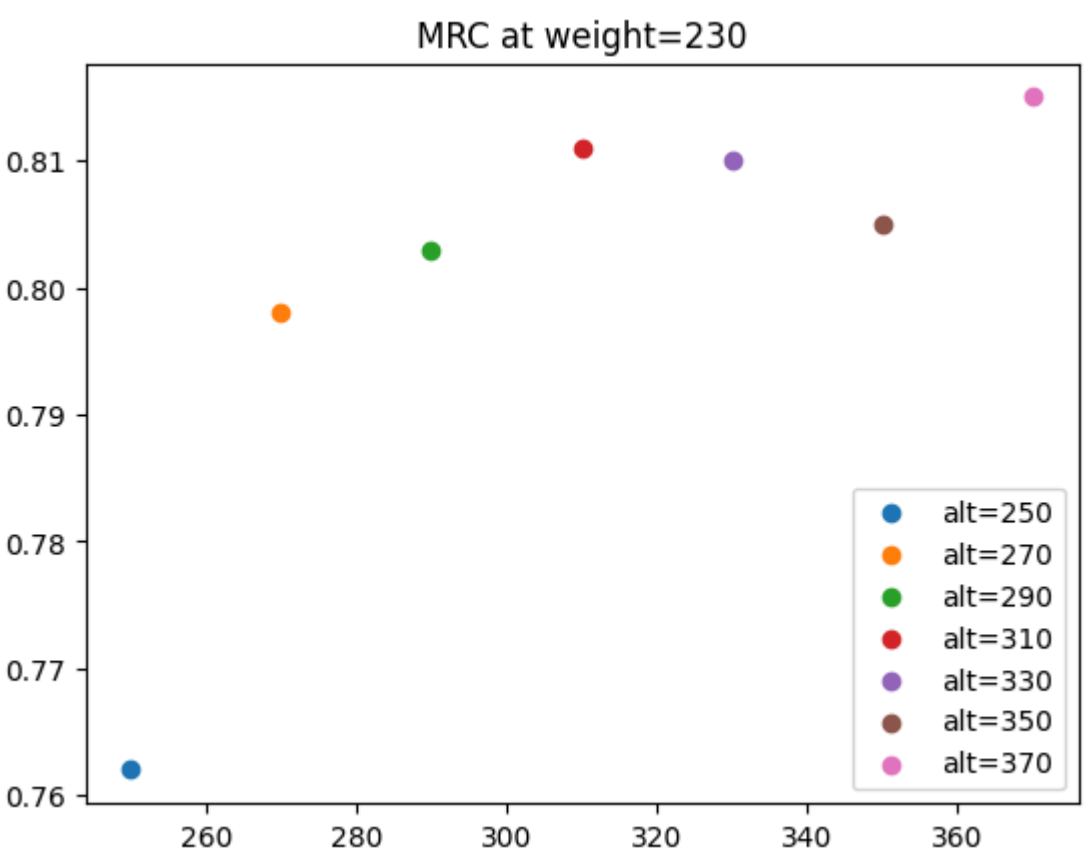

MRC at weight=180
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
alt=370
alt=390
alt=410
alt=430
0.800
0.775
0.750
0.725
0.700
0.675
0.650
0.625
250
275
300
325
350
375
400
425

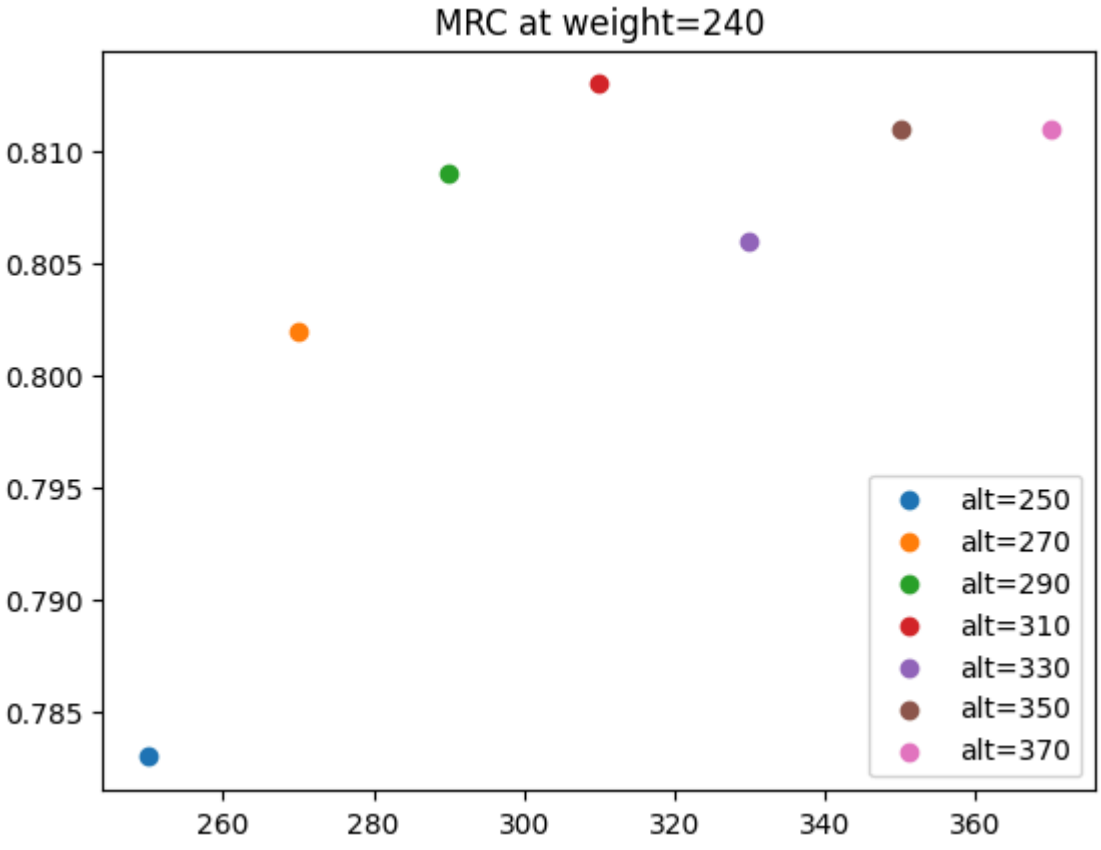

MRC at weight=190
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
alt=370
alt=390
alt=410
0.800
0.775
0.750
0.725
0.700
0.675
0.650
260
280
300
320
340
360
380
400

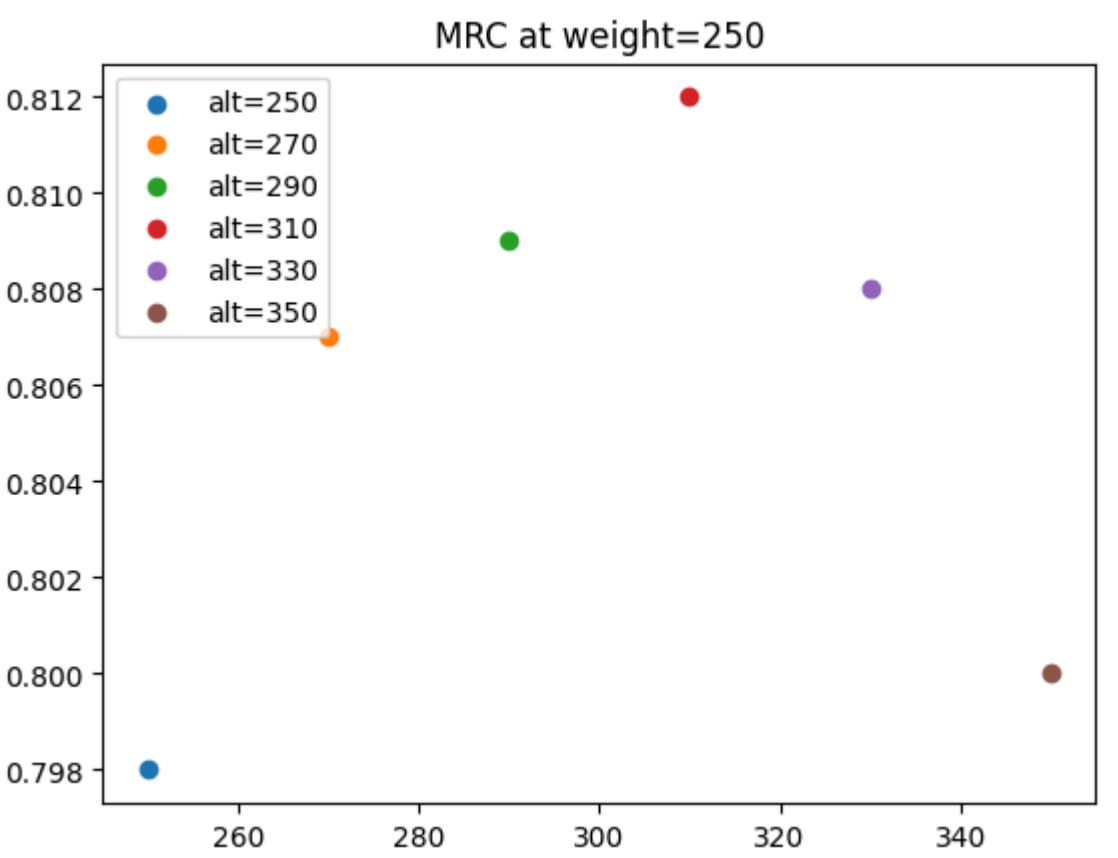

MRC at weight=200
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
alt=370
alt=390
alt=410
0.82
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.66
260
280
300
320
340
360
380
400

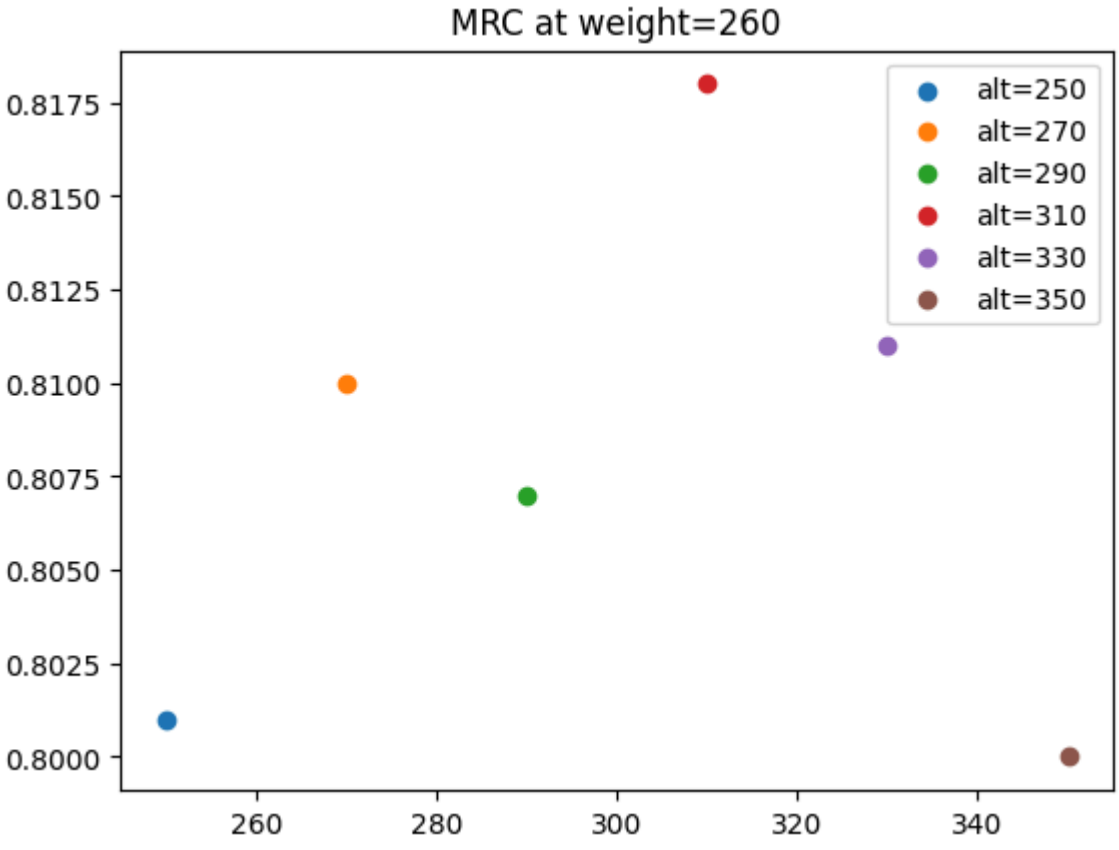

MRC at weight=210
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
alt=370
alt=390
0.82
0.80
0.78
0.76
0.74
0.72
0.70
260
280
300
320
340
360
380

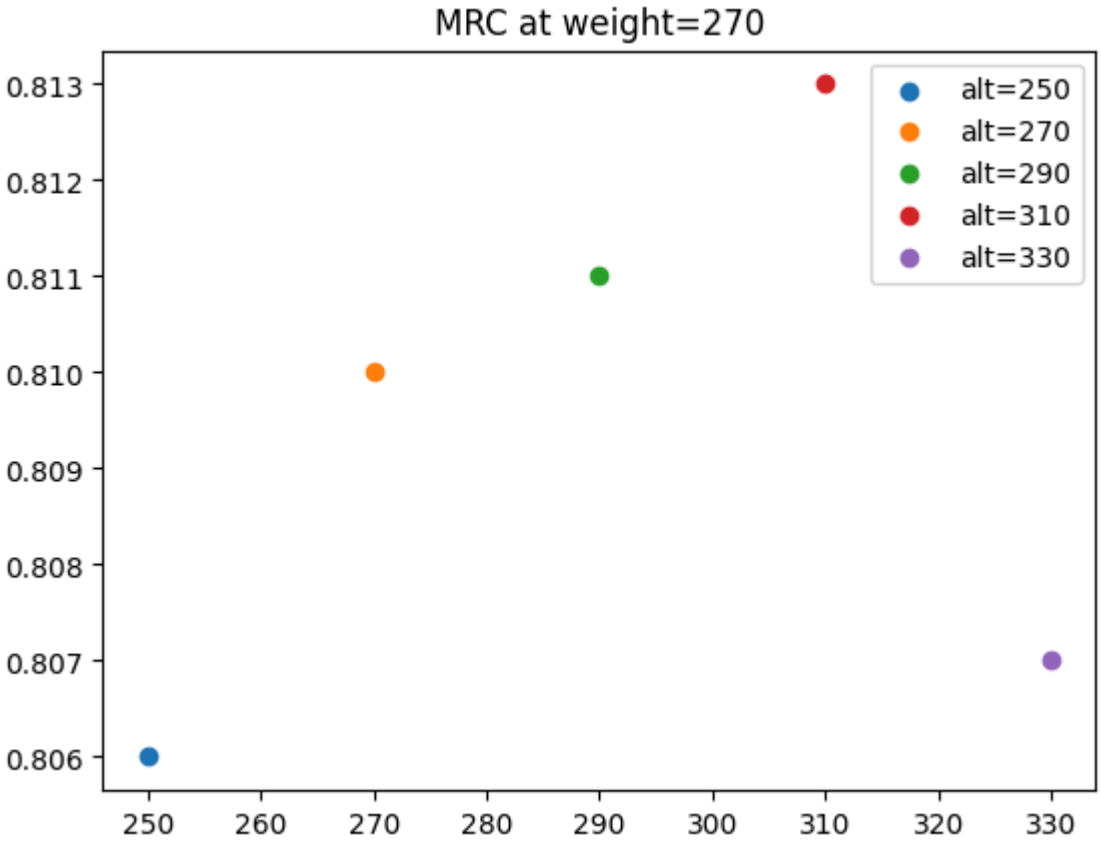

MRC at weight=220
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
alt=370
0.81
0.80
0.79
0.78
0.77
0.76
0.75
0.74
0.73
260
280
300
320
340
360

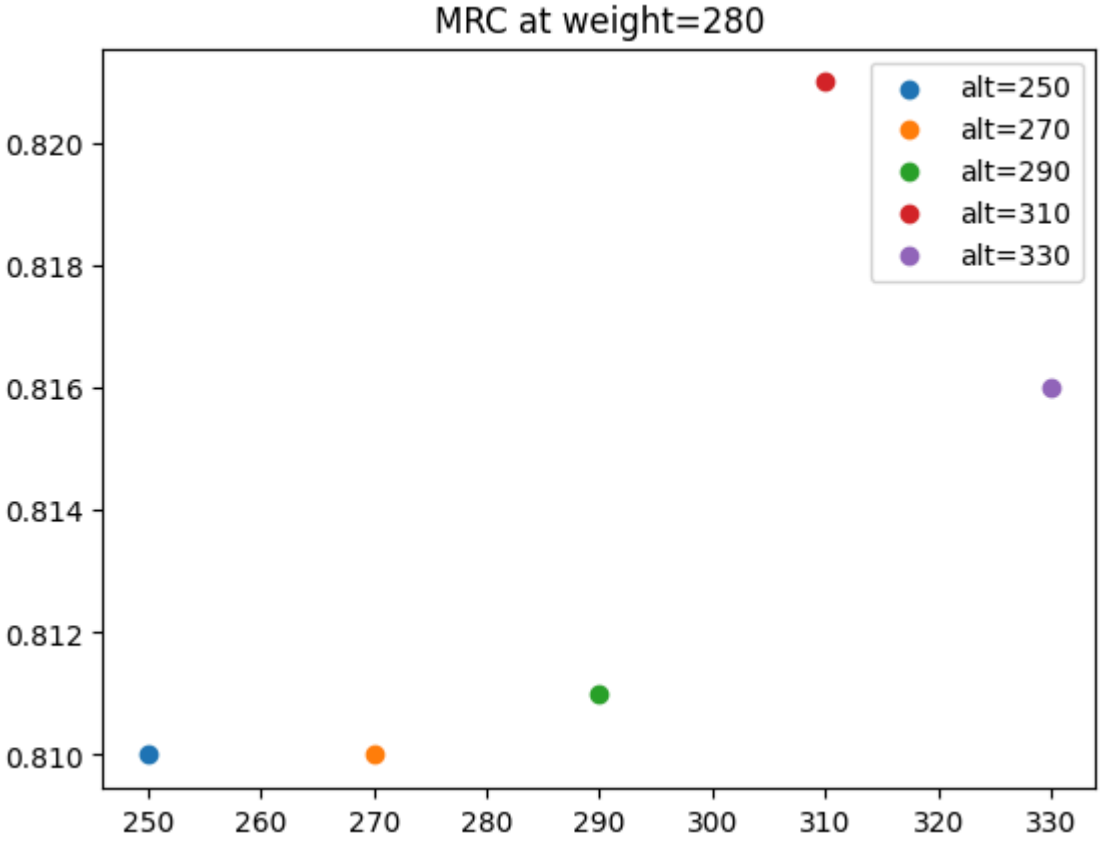

MRC at weight=230
0.81
0.80
0.79
0.78
0.77
0.76
260
280
300
320
340
360
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
alt=370

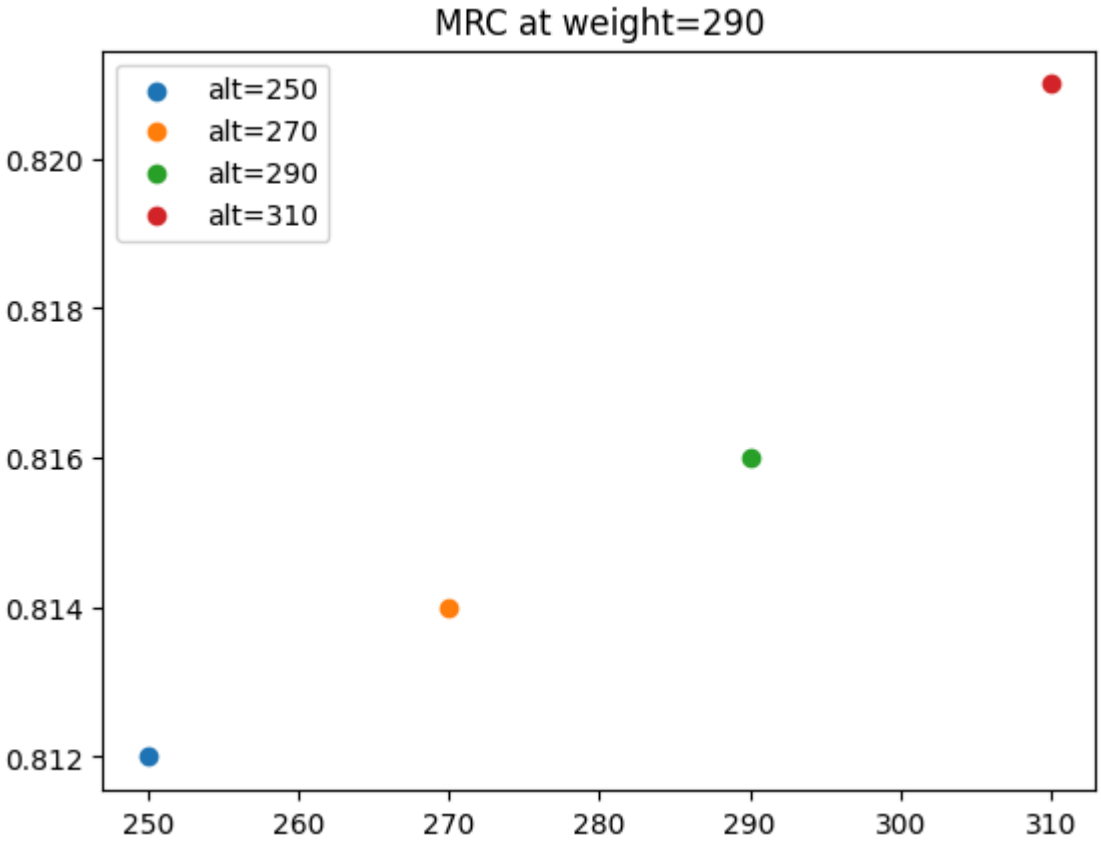

MRC at weight=240
0.810
0.805
0.800
0.795
0.790
0.785
260
280
300
320
340
360
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
alt=370


MRC at weight=250
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
0.812
0.810
0.808
0.806
0.804
0.802
0.800
0.798
260
280
300
320
340


MRC at weight=260
alt=250
alt=270
alt=290
alt=310
alt=330
alt=350
0.8175
0.8150
0.8125
0.8100
0.8075
0.8050
0.8025
0.8000
260
280
300
320
340


MRC at weight=270
0.813
0.812
0.811
0.810
0.809
0.808
0.807
0.806
250
260
270
280
290
300
310
320
330
alt=250
alt=270
alt=290
alt=310
alt=330


MRC at weight=280
0.820
0.818
0.816
0.814
0.812
0.810
250
260
270
280
290
300
310
320
330
alt=250
alt=270
alt=290
alt=310
alt=330



In [ ]:
```
for alt in range(10):
  file = open(f"ci2mach_{250 + alt * 20}.data", "w")
  for weight in range(16):
    try:
      file.writelines(",".join(str(round(x, 3)) for x in curves[alt][weight].tolist()) + "\n"
    except IndexError:
      pass
```

In [ ]:
```
alt = 290
weight = 250
ci = 80

print(int((alt - 250) / 20))
print(int((weight - 140) / 10))
print(curves[int((alt - 250) / 10)][int((weight - 140) / 10)][ci])
```

```
2
11
0.8266212280043118
```