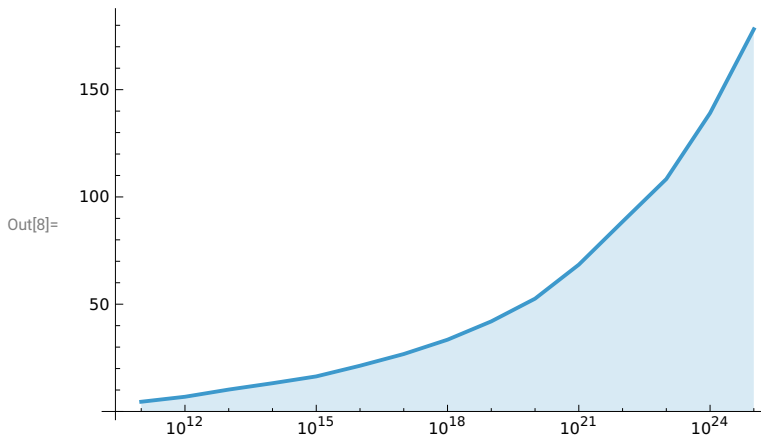


```
In[4]:= (* List of fast Gourdon alpha factors (alpha = alpha_y * alpha_z) found by
        running pi(x) benchmarks using the find_optimal_alpha_gourdon.sh script *)
```

```
In[5]:= alphaGourdon = {{10 ^ 11, 4.5278}, {10 ^ 12, 6.8509}, {10 ^ 13, 10.206},
                        {10 ^ 14, 13.171}, {10 ^ 15, 16.342}, {10 ^ 16, 21.316}, {10 ^ 17, 26.776},
                        {10 ^ 18, 33.448}, {10 ^ 19, 41.991}, {10 ^ 20, 52.580}, {10 ^ 21, 68.432},
                        {10 ^ 22, 88.472}, {10 ^ 23, 108.347}, {10 ^ 24, 139.079}, {10 ^ 25, 178.1645}}
```

```
Out[5]= {{100 000 000 000, 4.5278}, {1 000 000 000 000, 6.8509}, {10 000 000 000 000, 10.206},
          {100 000 000 000 000, 13.171}, {1 000 000 000 000 000, 16.342},
          {10 000 000 000 000 000, 21.316}, {100 000 000 000 000 000, 26.776},
          {1 000 000 000 000 000 000, 33.448}, {10 000 000 000 000 000 000, 41.991},
          {100 000 000 000 000 000 000, 52.58}, {1 000 000 000 000 000 000 000, 68.432},
          {10 000 000 000 000 000 000 000, 88.472}, {100 000 000 000 000 000 000 000, 108.347},
          {1 000 000 000 000 000 000 000 000, 139.079}, {10 000 000 000 000 000 000 000 000, 178.165}}
```

```
In[8]:= ListLogLinearPlot[alphaGourdon, Filling -> Bottom, Joined -> True]
```



```
In[9]:= (* alpha is a tuning factor that balances the computation of the
        easy special leaves (A + C formulas) and the hard special leaves
        (D formula). The formula below is used in the file src/util.cpp
        to calculate a fast alpha factor for the computation of pi(x). *)
```

```
In[11]:= NonlinearModelFit[alphaGourdon,
                        a (Log[x]) ^ 3 + b (Log[x]) ^ 2 + c Log[x] + d, {a, b, c, d}, x]
```

```
Out[11]= FittedModel[ -289. + 25.7 <<1>> - <<1>> + 0.00789 Log[x]^3 ]
```