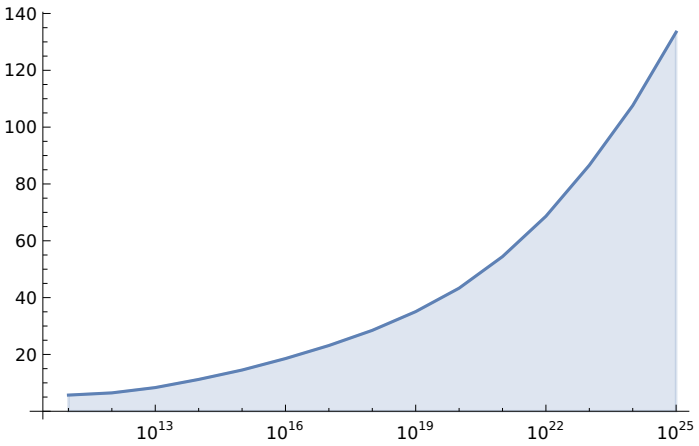
```
(* List of fast Gourdon alpha factors (alpha = alpha_y * alpha_z) found
 by running pi(x) benchmarks using the find_optimal_alpha_y.sh script *)
```

In[14]:=
```
alphaGourdon = {{10^11, 5.694}, {10^12, 6.470}, {10^13, 8.336},
  {10^14, 11.210}, {10^15, 14.516}, {10^16, 18.552}, {10^17, 23.143},
  {10^18, 28.479}, {10^19, 35.073}, {10^20, 43.327}, {10^21, 54.440},
  {10^22, 68.642}, {10^23, 86.600}, {10^24, 107.593}, {10^25, 133.439}}
```

Out[14]=
```
{{100 000 000 000, 5.694}, {1 000 000 000 000, 6.47}, {10 000 000 000 000, 8.336},
 {100 000 000 000 000, 11.21}, {1 000 000 000 000 000, 14.516},
 {10 000 000 000 000 000, 18.552}, {100 000 000 000 000 000, 23.143},
 {1 000 000 000 000 000 000, 28.479}, {10 000 000 000 000 000 000, 35.073},
 {100 000 000 000 000 000 000, 43.327}, {1 000 000 000 000 000 000 000, 54.44},
 {10 000 000 000 000 000 000 000, 68.642}, {100 000 000 000 000 000 000 000, 86.6},
 {1 000 000 000 000 000 000 000 000, 107.593}, {10 000 000 000 000 000 000 000 000, 133.439}}
```

In[15]:=
```
ListLogLinearPlot[alphaGourdon, Filling → Bottom, Joined → True]
```

Out[15]=




```
(* alpha is a tuning factor that balances the compuation 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[17]:=
```
NonlinearModelFit[alphaGourdon, a (Log[x])^3 + b (Log[x])^2 + c Log[x] + d, {a, b, c, d}, x]
```

Out[17]=
$$\text{FittedModel}\left[-167.085 + 15.2351\,\text{Log}[x] - 0.460441 \ll 1 \gg^2 + 0.00497225\,\text{Log}[x]^3\right]$$