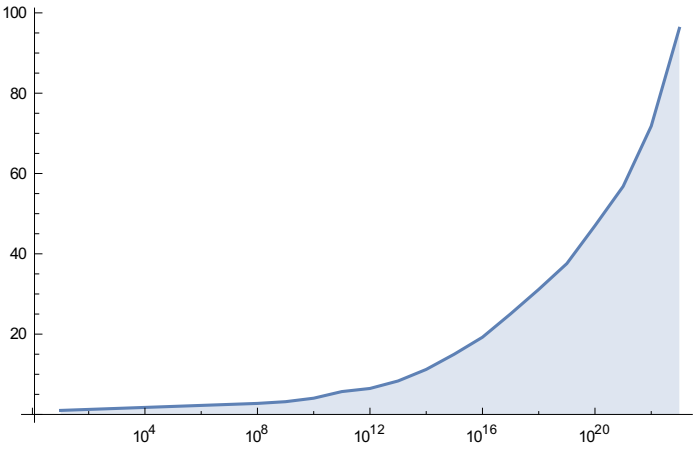
In[1]:=
```
(* List of fast Gourdon alpha factors (alpha = alpha_y * alpha_z) found
 by running pi(x) benchmarks using the find_optimal_alpha_y.sh script *)

alphaGourdon = { {10^1, 1}, {10^8, 2.746}, {10^9, 3.177}, {10^10, 4.047},
  {10^11, 5.694}, {10^12, 6.470}, {10^13, 8.336}, {10^14, 11.210},
  {10^15, 15.016}, {10^16, 19.231}, {10^17, 25.050}, {10^18, 31.139},
  {10^19, 37.573}, {10^20, 47}, {10^21, 56.783}, {10^22, 71.804}, {10^23, 96.146}}
```

Out[1]=
```
{{10, 1}, {100 000 000, 2.746}, {1 000 000 000, 3.177}, {10 000 000 000, 4.047}, {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, 15.016}, {10 000 000 000 000 000, 19.231}, {100 000 000 000 000 000, 25.05},
 {1 000 000 000 000 000 000, 31.139}, {10 000 000 000 000 000 000, 37.573},
 {100 000 000 000 000 000 000, 47}, {1 000 000 000 000 000 000 000, 56.783},
 {10 000 000 000 000 000 000 000, 71.804}, {100 000 000 000 000 000 000 000, 96.146}}
```

In[2]:=
```
ListLogLinearPlot[alphaGourdon, Filling → Bottom, Joined → True]
```

Out[2]=




```
(* 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/common.cpp to calculate a fast alpha factor for the computation of pi(x). *)

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

Out[3]=
$$\text{FittedModel}\left[-2.60402 + 1.51385\,\text{Log}[x] - 0.0958799\,\text{«1»}^2 + 0.00189716\,\text{Log}[x]^3\right]$$