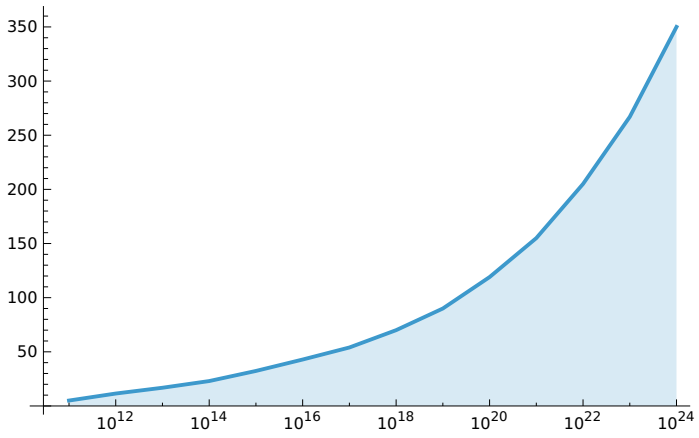
```
(* List of fast Gourdon alpha factors found by running pi
 (x) benchmarks using the find_fastest_alpha_gourdon.sh script *)
```

In[18]:=
```
alphaGourdon = {{10 ^ 11, 5}, {10 ^ 12, 11.475}, {10 ^ 13, 16.822}, {10^14, 23},
  {10^15, 32.337}, {10^16, 42.876}, {10^17, 54}, {10^18, 70}, {10^19, 90},
  {10^20, 119.022}, {10^21, 155}, {10^22, 205}, {10^23, 267}, {10^24, 350}}
```

Out[18]=
```
{{100 000 000 000, 5}, {1 000 000 000 000, 11.475}, {10 000 000 000 000, 16.822},
 {100 000 000 000 000, 23}, {1 000 000 000 000 000, 32.337}, {10 000 000 000 000 000, 42.876},
 {100 000 000 000 000 000, 54}, {1 000 000 000 000 000 000, 70},
 {10 000 000 000 000 000 000, 90}, {100 000 000 000 000 000 000, 119.022},
 {1 000 000 000 000 000 000 000, 155}, {10 000 000 000 000 000 000 000, 205},
 {100 000 000 000 000 000 000 000, 267}, {1 000 000 000 000 000 000 000 000, 350}}
```

In[19]:=
```
ListLogLinearPlot[alphaGourdon, Filling → Bottom, Joined → True]
```

Out[19]=



In[9]:=
```
(* 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[20]:=
```
NonlinearModelFit[alphaGourdon,
 a (Log[x])^3 + b (Log[x])^2 + c Log[x] + d, {a, b, c, d}, x]
```

Out[20]=

FittedModel[$-758. + 68. \ll 1 \gg - \ll 1 \gg + 0.0211\,\text{Log}[x]^3$]