marxer engineering & computing

www.mec.li

Mathematica und Thin Film Design

Mathematica ist auch ein sehr leistungsstarkes Programmier Werkzeug. In diesem Beispiel soll gezeigt werden, dass für das Design eines Anti Reflection Coatins (ARC) nur sehr wenig Code notwendig ist, und die eingebauten Plot Funktionen die Ergebnisse anschaulich darstellen lassen.

Bemerkung: mec hat auch Berechnungen für kompliziertere Schichtsysteme durchgeführt. In Anlehnung an das Paper "Coating design contest: Antireflection coating for lenses to be used with normal and infrared photographic film" von Alfred Thelen und Roland Langfeld (SPIE Vol. 1782 Thin Films for Optical Systems (1992)). Für diese Analyse wuden die Berechnungen mit charakteristischen Matrizen durchgeführt und einige zeitkritische Teile kompiliert. Auf diese Weise war es möglich, die im Wettbewerb mit der weltweit besten Thin Film Software erzielten Ergebnisse zu reproduzieren, für bestimmte Dicken sogar bessere Lösungen zu finden.

Aufgabe - Design eines Antireflection Coating

Gegeben seien zwei Materialien mit Brechwert 1.38 und 2.25. Auf Glas soll eine reflexmindernde Schicht aufgebracht werden, so dass die Reflexion im Wellenlängenbereich von 400nm bis 900nm möglichst klein ist.

Theorie

Es gibt verschiedene Ansätze zur Berechnung der optischen Interferenzen an dünnen Filmen. Im Folgenden wird die Berechnung unter Verwendung der Fresnel Reflection Koeffizienten und der Summe für die multiplen Reflexionen an den Schichtübergängen verwendet, da damit eine einfache Formulierung mittels Rekursion gezeigt werden kann. Die Gleichungen gelten für Reflexion und senkrechten Einfall des Lichts. Weitere Details wie Dispersion werden in dieser einfachen Betrachtung nicht berücksichtigt.

```
rFre[i_, j_] := (n[j] - n[i])/(n[j] + n[i]); (* Fresnel Reflection Coefficient *)
β[i_] := (4 Pi I t[i] n[i])/λ;  (* Phasenänderung *)
r[i_] := (rFre[i, i + 1] + r[i - 1] E^β[i])/(1 + rFre[i, i + 1] r[i - 1] E^β[i]); (* Aufsummieren der Reflexionen *)
r[0] := rFre[0, 1];
```

Input

Input und Definition der Merit Funktion

```
Clear[n, nH, nL, nm, λMin, λMax, λPoints,
  λDel, merit, t, m, mBest, sol, t1Start, t2Start];
n[0] = 1.5; nH = 2.25; nL = 1.38; n[3] = 1.0;
nm = 10^-9; λMin = 400 nm; λMax = 900 nm;
λPoints = 100; λDel = (λMax - λMin)/(λPoints - 1); (* λ Bereich *)
merit[films_] := Module[{s}, s = 0;
  Do[λ = λMin + (i - 1) λDel; s = AddTo[s, Abs[r[films]]^2], {i, λPoints}]; s];
```

Berechnung

Monte Carlo Suche: In einem ersten Schritt wird mittels einer Monte Carlo Suche ermittelt, welches Material innen und welches Material aussen liegen soll. Gleichzeitig werden auch Startwerte für die Schichtdicken für die anschliessende genauere Minimumsuche bestimmt ...

```
Do[
  If[Random[Integer] == 0, n[1] = nL; n[2] = nH, n[1] = nH; n[2] = nL];
  t[1] = Random[Real, {0, 200 nm}];
  t[2] = Random[Real, {0, 200 nm}];
  m = merit[2];
  If[Or[i == 1, m < mBest], mBest = m; solMC = {mBest, n[1], t[1], n[2], t[2]}];
  , {i, 400}];
solMC
```

```
{1.7281, 2.25, 1.00372×10^-8, 1.38, 1.40867×10^-7}
```

Optimierung: Mit diesen Startwerten wird mittels NMinimize[] die beste Lösung ermittelt. ...

```
{dummy, n[1], t1Start, n[2], t2Start} = solMC;
 sol = NMinimize[t[1] = t1; t[2] = t2; merit[2],
   {{t1, 0.8 t1Start, 1.2 t1Start}, {t2, 0.8 t2Start, 1.2 t2Start}}];
 ({sol[[1]], n[1], t1, n[2], t2}) /. sol[[2]]
```

```
{1.53219, 2.25, 6.51483×10^-9, 1.38, 1.3018×10^-7}
```

Mit wenig Code resultiert das Ergebnis: wenn zuerst 6.5nm hochbrechendes Material und dann 130nm tiefbrechendes Material aufgedampft wird, resultiert das beste ARC Verhalten im Bereich von 400nm bis 900nm.

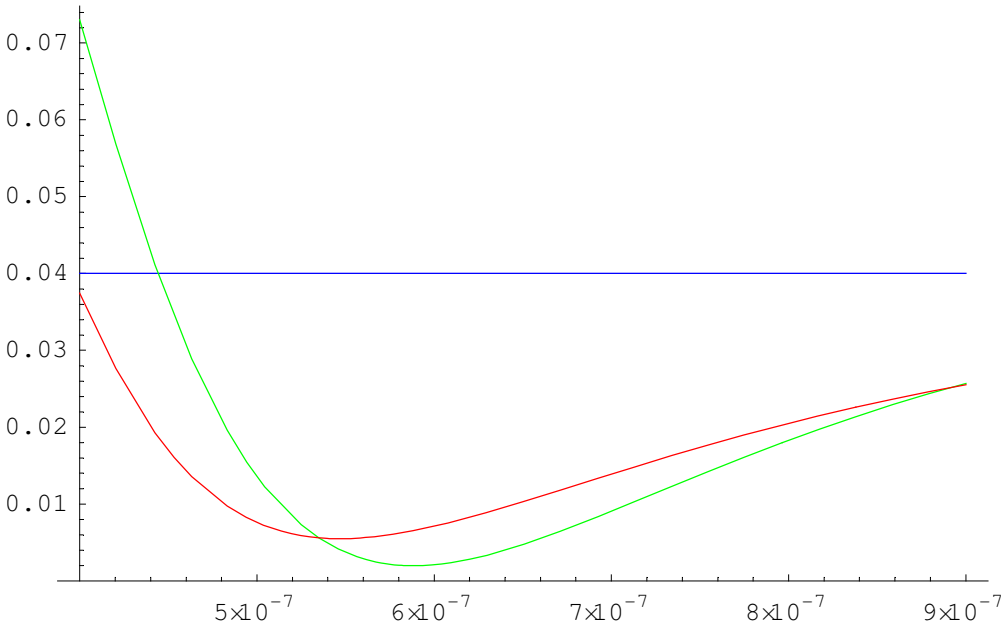
Report und Plot der Ergebnisse

Zum Abschluss werden noch die Spektren für die 3 Situationen

- keine Schicht (blau)
- von MonteCarlo Simulation gelieferte Schichtdicken (grün)
- von NMinimize gelieferte Schichtdicken (rot)

dargestellt.

```
gr1 = Plot[(λ = λVar; Abs[rFre[0, 3]]^2), {λVar, 400 nm, 900 nm},
   PlotStyle → {RGBColor[0, 0, 1]},
   PlotRange → {0, Automatic}, DisplayFunction → Identity];
{merit, n[1], t[1], n[2], t[2]} = solMC;
gr2 = Plot[(λ = λVar; Abs[r[2]]^2), {λVar, 400 nm, 900 nm},
       PlotRange → {0, Automatic},
   PlotStyle → {RGBColor[0, 1, 0]}, DisplayFunction → Identity];
({t[1], t[2]} = {t1, t2}) /. sol[[2]];
gr3 = Plot[(λ = λVar; Abs[r[2]]^2), {λVar, 400 nm, 900 nm},
       PlotRange → {0, Automatic},
   PlotStyle → {RGBColor[1, 0, 0]}, DisplayFunction → Identity];
Show[{gr1, gr2, gr3}, DisplayFunction → $DisplayFunction,
  Background → RGBColor[1, 1, 1]];
```



Die Graphik zeigt, dass die Reflexion von 4% (in blau) stark verbessert werden kann. Die Monte Carlo Simulationen (in grün) ergeben bei jedem Run unterschiedliche Startwerte, die die Reflexion von 4% jedoch schon deutlich reduzieren. Die Optimierung mit NMinizize[] (in rot) liefert mit diesen Startwerten jeweils die gleiche und für diese Merit Funktion optimale Lösung.

Es versteht sich von selbst, dass mit wenig Aufwand der Output noch verschönert und ein umfassender Report generiert werden könnte.