Perturbative approach to Separated Eigenvalues Instability Criterion

```
$Assumptions = {Element[i11 , Reals] &&
    Element[i10, Reals] && Element[i00, Reals] && Element[a11, Reals] &&
    Element[a01, Reals] && Element[a10, Reals] && Element[a00, Reals] };

F = ({
 {i11, i10},
 {i10, i00}
});
A = ({
 {I a11, I a01},
 {I a10, I a00}
});
Λ = Inverse[F] . A;
Λ // MatrixForm
FullSimplify[Eigenvalues[Λ ]]
S = Tr[Λ] // Simplify;
DetL = (Tr[Λ]) ^ 2 - 4 Det[Λ] ;
```

check for $\epsilon = 0$

```
Fe0 = ({
 {n11^2, 0},
 {0, n00^2}
});
Ae0 = ({
 {I w n11^2, 0},
 {0, I w n00^2}
});
Λe0 = Inverse[Fe0] . Ae0;
Λe0 // MatrixForm
```

Expansion in ϵ and δ

```
i11 = n11^2 + d1 δ;
i10 = γ1 ϵ + γ ϵ^2;
i00 = n00^2 + d0 δ;
a11 = w n11^2 + a1 δ + α1 ϵ^2;
a10 = β1 ϵ;
a01 = β0 ϵ;
a00 = w n00^2 + a0 δ + α0 ϵ^2;
DetLd0 = DetL /. δ → 0;
Series[DetLd0, {ϵ, 0, 2}]
```

```
Series[DetLd0, {ϵ, 0, 2}] // Simplify
```

Entanglement Coefficients

```
X = {x, y}; grad[f_] := D[f, {X}]; perp[{k1_, k2_}] := {k2, -k1};
k0 = {m, n}; c = N m / (m^2 + n^2) ^ (3 / 2) k0; μ = {μ1, μ2}; zero = {0, 0};
Ω[{k1_, k2_}] := N * k1 / (k1^2 + k2^2) ^ (1 / 2);
d[{k1_, k2_}] := ( ( (k1 + μ1) ^2 + (k2 + μ2) ^2) Ω[{k1, k2} + μ] ) ^ -1;
w[{k1_, k2_}, σ_] := c . ({k1, k2} + μ) + σ Ω[{k1, k2} + μ];
la[{k1_, k2_}, σ_] := -ν * ((k1 + μ1) ^2 + (k2 + μ2) ^2) + I * w[{k1, k2}, σ];
R[j_, j1_][σ_, σ1_] :=
  (-I) * Residue[1 / ((z - la[j, σ]) (z - la[j1, σ1])), {z, la[j, σ]}] ;
f[{j1_, j2_}, σ_] := Exp[I {j1, j2} . X] / Sqrt[2] {1, -σ };
(* eigenvectors in the new variables *)
fHarm[{j1_, j2_}, σ_] := Exp[-I {j1, j2} . X] f[{j1, j2}, σ];

Jmubar[{k1_, k2_}] := {{0, I (k1 + μ1)}, {I (k1 + μ1), 0}};

L00Harm[{kappa1_, kappa2_}][{k1_, k2_}] := KroneckerDelta[0, kappa1]
   KroneckerDelta[0, kappa2] {{-ν * ((k1 + μ1) ^2 + (k2 + μ2) ^2) + I c . ({k1, k2} + μ) ,
     -N * I ( k1 + μ1) / Sqrt[(k1 + μ1) ^2 + (k2 + μ2) ^2] },
    { -N * I ( k1 + μ1) / Sqrt[(k1 + μ1) ^2 + (k2 + μ2) ^2] ,
     -ν * ((k1 + μ1) ^2 + (k2 + μ2) ^2) + I c . ({k1, k2} + μ) }};
L01Harm[{kappa1_, kappa2_}][{k1_, k2_}] :=
  ϵ / 2 (KroneckerDelta[m, kappa1] KroneckerDelta[n, kappa2] +
     KroneckerDelta[-m, kappa1] KroneckerDelta[-n, kappa2])
   { {  -( (m^2 + n^2) ^ (1 / 2) ) ^ -1 I perp[k0] . ({k1, k2} + μ) ,
     I perp[k0] . (({k1, k2} + μ) / Sqrt[(k1 + μ1) ^2 + (k2 + μ2) ^2]) } ,
    {0,  1 / Sqrt[(k1 + kappa1 + μ1) ^2 + (k2 + kappa2 + μ2) ^2] *
      (( -(m^2 + n^2) ^ (1 / 2) ) ^ -1 I perp[k0] . ({k1, k2} + μ) *
         Sqrt[(k1 + μ1) ^2 + (k2 + μ2) ^2] + (m^2 + n^2) ^ (1 / 2) I
         perp[k0] . (({k1, k2} + μ) / Sqrt[(k1 + μ1) ^2 + (k2 + μ2) ^2])) } };
L0Harm[ell_][{kappa1_, kappa2_}][{k1_, k2_}] :=
  If[ ell == 0, L00Harm[{kappa1, kappa2}][{k1, k2}] ,
   If[ ell == 1, L01Harm[{kappa1, kappa2}][{k1, k2}] , 0] ];

B[ell_][{k1_, k2_}][{jp1_, jp2_}, σp_][{j1_, j2_}, σ_] :=
  If[ {j1 + k1, j2 + k2} == {jp1, jp2},
   (  Jmubar[{j1 + k1, j2 + k2}] . (L0Harm[ell][{k1, k2}][{j1, j2}] .
        fHarm[{j1, j2}, σ] ) ) . fHarm[{jp1, jp2}, σp] , 0];

L[ell_][{k1_, k2_}][{jp1_, jp2_}, σp_][{j1_, j2_}, σ_] :=
  If[ {j1 + k1, j2 + k2} == {jp1, jp2},
   ( L0Harm[ell][{k1, k2}][{j1, j2}] . fHarm[{j1, j2}, σ] ) .
    fHarm[{jp1, jp2}, σp] , 0];


{x, y}
```

```
Residue[1 / ((z - I a) (z - I b)), {z, I a}] // FullSimplify
```

Test

```
d[{k1, k2}]
d[zero]
w[zero, 1] // FullSimplify
w[k0, -1] // FullSimplify
w[{j1, j2}, 1] // FullSimplify
R[zero, {j1, j2}][1, 1] // FullSimplify
```

```
L00Harm[{kappa1, kappa2}][{k1, k2}] // MatrixForm
L01Harm[{kappa1, kappa2}][{k1, k2}] // MatrixForm
Jmubar[{k1, k2}] . L00Harm[{kappa1, kappa2}][{k1, k2}] // Simplify //
 MatrixForm
Jmubar[{k1, k2}] . L01Harm[{kappa1, kappa2}][{k1, k2}] // Simplify // MatrixForm
```

```
Simplify[ MatrixForm[ L00Harm[zero][{k1, k2}] ], Assumptions → {m ≠ 0, n ≠ 0} ]
Simplify[ MatrixForm[ L01Harm[k0][{k1, k2}] ], Assumptions → {m ≠ 0, n ≠ 0} ]
```

```
Simplify[ MatrixForm[ L01Harm[-k0][{k1, k2}] ], Assumptions → {m ≠ 0, n ≠ 0} ]
```

```
Simplify[ L01Harm[-k0][k0] . fHarm[k0, -1], Assumptions → {m ≠ 0, n ≠ 0} ]
```

```
FullSimplify[ Jmubar[zero] . (L01Harm[-k0][k0] . fHarm[k0, -1]),
 Assumptions → {m ≠ 0, n ≠ 0} ]
```

```
B1mk = FullSimplify[ B[1][-k0][zero, -1][k0, -1] , Assumptions → {m ≠ 0, n ≠ 0} ]
 (* checked also with pen and paper *)
```

```
B1pk = FullSimplify[ B[1][k0][k0, 1][zero, 1] , Assumptions → {m ≠ 0, n ≠ 0} ]
```

```
L000pm =
 FullSimplify[ L[0][zero][zero, 1][zero, -1] , Assumptions → {m ≠ 0, n ≠ 0} ]
```

```
L000mp =
 FullSimplify[ L[0][zero][zero, -1][zero, 1] , Assumptions → {m ≠ 0, n ≠ 0} ]
```

```
L0kkpm = FullSimplify[ L[0][zero][k0, 1][k0, -1] , Assumptions → {m ≠ 0, n ≠ 0} ]
```

```
L0kkmp = FullSimplify[ L[0][zero][k0, -1][k0, 1] , Assumptions → {m ≠ 0, n ≠ 0} ]
```

```
L1k0mp = FullSimplify[ L[1][k0][k0, -1][zero, 1] , Assumptions → {m ≠ 0, n ≠ 0} ]
Simplify[L1k0mp]
```

```
L10kpm = FullSimplify[ L[1][-k0][zero, 1][k0, -1] , Assumptions → {m ≠ 0, n ≠ 0} ]
```

Computation of ϵ Subscript[γ, 1]

```
fHarm[zero, -1] . fHarm[zero, +1]
fHarm[k0, -1] . fHarm[k0, +1]
```

```
1 / R[zero, zero][+1, -1] // FullSimplify
1 / R[k0, k0][-1, 1]
```

```
(* d[zero]  B1mk R[k0,zero][-1,-1] (   fHarm[zero,-1] .fHarm[zero,+1] )-
 d[k0]  B1pk R[zero,k0][1,1]  (   fHarm[k0,-1] .fHarm[k0,+1] ) *)
```

```
epsg1 = FullSimplify[
d[zero] B1mk R[zero, zero][1, -1] ( fHarm[zero, -1] . fHarm[zero, +1] ) -
   d[k0] B1pk R[k0, k0][-1, 1] ( fHarm[k0, -1] . fHarm[k0, +1] )
, Assumptions → {m ≠ 0, n ≠ 0} ]
g1 = epsg1 / ϵ;
```

Computation of i ϵ Subscript[β, 1]

```
(* L10kpm +d[zero]  B1 k R[zero,k0][1,1] L000pm-
 d[k0]  B1pk R[zero,k0][1,1]  L0kkpm *)
```

```
iepsb1 = Simplify[
L10kpm + d[zero] B1mk R[k0, k0][-1, 1] L000pm - d[k0] B1pk R[k0, k0][-1, 1] L0kkpm
, Assumptions → {m ≠ 0, n ≠ 0} ]
b1 = iepsb1 / (I ϵ);
```

Computation of i ϵ Subscript[β, 0]

```
(* L1k0mp -d[k0]  B1pk R[zero,k0][1,1] L0kkmp+
 d[zero]  B1mk R[k0,zero][-1,-1]  L000mp *)
```

```
iepsb0 = Simplify[
L1k0mp - d[k0] B1pk R[k0, k0][-1, 1] L0kkmp +
   d[zero] B1mk R[zero, zero][1, -1] L000mp
, Assumptions → {m ≠ 0, n ≠ 0} ]
b0 = iepsb0 / (I ϵ);
```

Instability Criterion: Application

We want to check that
-(1/(n00^2 n11^2))(β0 β1 - w (β0+β1) γ1+w^2 γ1^2) >0

```
- (1 / (n00^2 n11^2)) ( β0 β1 - w (β0 + β1) γ1 + w^2 γ1^2) // Simplify
```

```
N002 = fHarm[zero, +1] . fHarm[zero, +1]
N112 = fHarm[k0, -1] . fHarm[k0, -1]

Discr = -(1 / (N002 N112)) ( b0 - w [zero, +1] g1) (b1 - w [zero, +1] g1);
```

```
-I L000pm - w[zero, 1] ( fHarm[zero, -1] . fHarm[zero, +1]) // FullSimplify
```

```
-I L0kkpm - w[k0, -1] ( fHarm[k0, -1] . fHarm[k0, +1]) // FullSimplify
```

```
-I L000mp - w[zero, 1] ( fHarm[zero, -1] . fHarm[zero, +1]) // FullSimplify
```

```
-I L0kkmp - w[k0, -1] ( fHarm[k0, -1] . fHarm[k0, +1]) // FullSimplify
```

```
FullSimplify[-I L10kpm - 1 / 2 d[zero] B1mk ( mu1^2 + mu2^2 - N^2)]
```

```
FullSimplify[-I L1k0mp + 1 / 2 d[k0] B1pk ( (m + mu1) ^2 + (n + mu2) ^2 - N^2)]

mubar = {(m n^3) / (m^2 + n^2) ^ (3 / 2) τ^2, n τ};
DiscrPar = Discr /. {μ1 → mubar[[1]], μ2 → mubar[[2]]} ;
Series[ DiscrPar, {τ, 0, 1}, Assumptions → {τ > 0, n > 0}]
```

```
I[σ_] := If[σ == 1,
 perp[ k0] . mubar / (2 Ω[k0 + mubar] ( (k0 + mubar) . (k0 + mubar) ))
    (Ω[k0 + mubar] ( k0 . k0 - (mubar) . (mubar) ) +
      ( m + mubar[[1]] ) N ( Sqrt[k0 . k0] - Sqrt[mubar . mubar] ) ) ,
If[σ == -1,
 -perp[k0] . mubar / (2 Ω[mubar] ( mubar . mubar) )
     (Ω[mubar] ( k0 . k0 - (k0 + mubar) . (k0 + mubar) ) +
       mubar[[1]] N (Sqrt[k0 . k0] - Sqrt[(k0 + mubar) . (k0 + mubar)]) )
, 0] ]

Series[ I[1], {τ, 0, 1}, Assumptions → {τ > 0}] // FullSimplify
Series[ I[-1], {τ, 0, 0}, Assumptions → {τ > 0}] // FullSimplify
```

Computation of the discriminant for large Floquet parameters

```
mupar = {m / (3 m^2 + 4 n^2) ^ (1 / 2) (n / 2 + 1 / τ) - m / 2, 1 / τ};
DiscrLargePar = Discr /. {μ1 → mupar[[1]], μ2 → mupar[[2]]} ;


(* Series[ DiscrLargePar,{τ,0,0}, Assumptions→ {τ>0,N>0,m>0,n>0}] *)
```