

```

In[2]:= ClearAll["Global`*"];
borra todo

$MaxExtraPrecision = 100;
máxima precisión extra

Potential[r_, l_, m_] := ((1 - 2/r) (1 + l^2 + 2/r + m^2 r^2)) / r^2;

FindRmax[l_?NumericQ, m_?NumericQ, prec_:50] := Module[{rMax},
    expresión numérica? expresión numérica? módulo
    rMax = r /. FindRoot[D[Potential[r, l, m], r] == 0, {r, 3}, WorkingPrecision → prec];
    encuentra... deriva precisión operativa

SetPrecision[rMax, prec];
asigna precisión

lambda0[u_, l_, m_, ω_] := Module[{k = Sqrt[ω^2 - m^2]},
    módulo raíz cuadrada
    ((-1 + 4 I ω) + u (4 - 3 u + 4 I (-2 + u) ω - 8 I k + 4 I u k)) / ((-1 + u)^2 u)];
    número i número i número i número i

s0[u_, l_, m_, ω_] := Module[{k = Sqrt[ω^2 - m^2]},
    módulo raíz cuadrada
    (1 (-1 + u) + l^2 (-1 + u) - 4 m^2 (1 + (-3 + u) u) + (-1 + u)
     (1 - 4 I k - 4 ω (I + 4 ω + 4 k) + u (-1 + 4 I k + 4 ω (I + 2 ω + 2 k)))) / ((-1 + u)^3 u)];
    número i número i número i número i

AIMIterateNumeric[c0_List, d0_List, Imax_Integer, Nmax_Integer] :=
    lista lista entero entero
    Module[{cMat, dMat, n, i, delta}, cMat = ConstantArray[0, {Nmax + 1, Imax + 2}];
    módulo arreglo constante
    dMat = ConstantArray[0, {Nmax + 1, Imax + 2}];
    arreglo constante
    cMat[[1, 1 ;; Imax + 1]] = c0;
    dMat[[1, 1 ;; Imax + 1]] = d0;
    For[n = 1, n ≤ Nmax, n++, For[i = 0, i ≤ Imax, i++, cMat[[n + 1, i + 1]] =
    para cada para cada
        (i + 1) cMat[[n, i + 2]] + dMat[[n, i + 1]] + Sum[c0[[k + 1]] × cMat[[n, i - k + 1]], {k, 0, i}]];
        suma
        dMat[[n + 1, i + 1]] = (i + 1) dMat[[n, i + 2]] +
        Sum[d0[[k + 1]] × cMat[[n, i - k + 1]], {k, 0, i}];];
    suma
    delta = dMat[[Nmax + 1, 1]] × cMat[[Nmax, 1]] - dMat[[Nmax, 1]] × cMat[[Nmax + 1, 1]]; delta];
    (*Delta AIM*)

DeltaNumeric[l_?NumericQ, m_?NumericQ, ω_?NumericQ, uMax_, Imax_Integer,
    expresión numérica? expresión numérica? expresión numérica? entero
    Nmax_Integer, prec_:50] := Module[{seriesLambda, seriesS, c0, d0},
    entero módulo
    seriesLambda =
        Collect[Normal@Series[lambda0[u, l, m, ω], {u, uMax, Imax}], u - uMax];
        agrupa ... normal serie
    seriesS = Collect[Normal@Series[s0[u, l, m, ω], {u, uMax, Imax}], u - uMax];
        agrupa ... normal serie
    c0 = Table[Coefficient[seriesLambda, u - uMax, i], {i, 0, Imax}];
        tabla coeficiente
    d0 = Table[Coefficient[seriesS, u - uMax, i], {i, 0, Imax}];
        tabla coeficiente

SetPrecision[AIMIterateNumeric[
    asigna precisión

```

```

SetPrecision[c0, prec], SetPrecision[d0, prec], Imax, Nmax], prec]];
Lasigna precisión Lasigna precisión

FindQNM[l_?NumericQ, m_?NumericQ, ωGuess_?NumericQ,
Lexpresión numérica? Lexpresión numérica? Lexpresión numérica?

Imax_ : 25, Nmax_ : 25, prec_ : 80] := Module[{rMax, uMax},
módulo

rMax = FindRmax[l, m, prec];
uMax = SetPrecision[1 - 2 / rMax, prec];
Lasigna precisión

FindRoot[DeltaNumeric[l, m, ω, uMax, Imax, Nmax, prec] == 0,
Lencuentra raíz

{ω, ωGuess}, WorkingPrecision → prec, AccuracyGoal → Floor[prec / 2],
Lprecisión operativa Lobjetivo de exactitud Lentero inferior

PrecisionGoal → Floor[prec / 2], MaxIterations → 200]
Lobjetivo de precisión Lentero inferior Lmáximo de iteraciones

```

```
In[1]:= {time, sol} = AbsoluteTiming[FindQNM[0, 0, 11 / 100 - 10 / 100 I, 60, 60, 60]]
Lduración absoluta Lnúmero i
```

```
Out[1]= {17.9592, {ω → 0.110454382218180011659275979049160772880575459453726315931789 -
0.104894533503689255914072315149419767238676773804936336190662 I}}
```

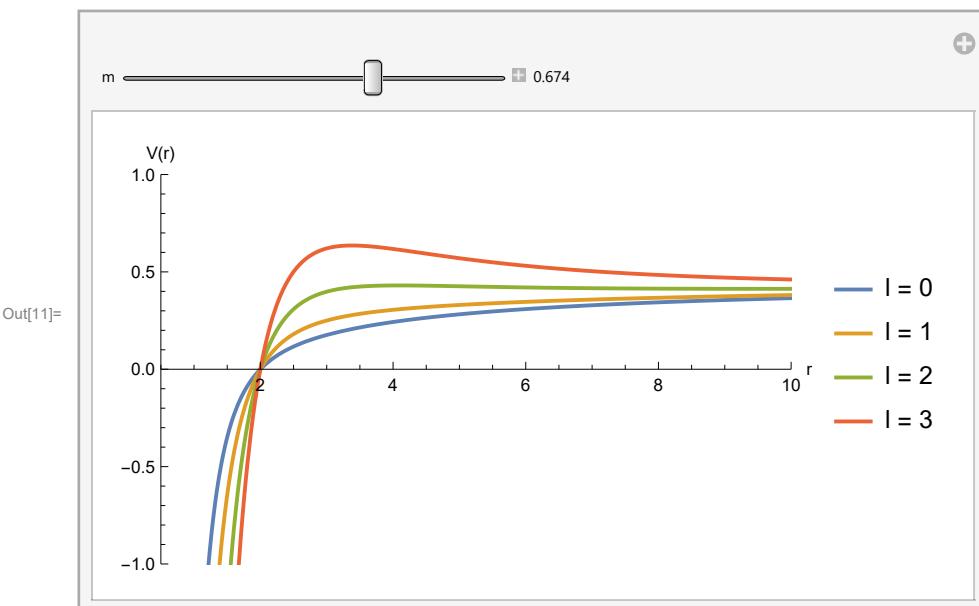
```
In[2]:= (*https://arxiv.org/pdf/2206.03512.pdf Comparar valores*)
```

```
In[11]:= lvals = {0, 1, 2, 3};
Manipulate[Plot[Evaluate[Table[Potential[r, l, m], {l, lvals}]],
Lmanipula Lrepräsentación Levalúa Ltabla

{r, 0.5, 10}, PlotRange → {{0.5, 10}, {-1, 1}}, PlotStyle → Thick,
Lrango de representación Lestilo de representación Lgrueso

AxesLabel → {"r", "V(r)"}, PlotLegends → ("l = " <> ToString[#[#] & /@ lvals]),
Letiqueta de ejes Lleyendas de representación Lconvierte a cadena de caracteres

{{m, 0.1, "m"}, 0, 1, Appearance → "Labeled"}]
Lapariencia Letiquetado
```



```
(*Guess con WKB 1º*)
QNWKBl_?NumericQ, m_?NumericQ, n_Integer : 0, prec_ : 50] :=
  [expresión numérica] [expresión] [entero]

Module[{r0, pot0, d2pot0, omega2, omega}, r0 = FindRmax[l, m, prec];
  [módulo]

  pot0 = Potential[r0, l, m];
  d2pot0 = (1 - 2 / r0) * D[(1 - 2 / r) * D[Potential[r, l, m], r], r] /. r → r0;
    [deriva] [deriva]

  omega2 = pot0 - I (n + 1 / 2) Sqrt[-2 d2pot0];
    [número i] [raíz cuadrada]

  omega = Sqrt[omega2];
    [raíz cuadrada]

  N[omega, prec]]
  [valor numérico]

In[]:= FindQNM1[l_?NumericQ, m_?NumericQ, n_Integer, Imax_ : 25, Nmax_ : 25, prec_ : 80] :=
  [expresión numérica] [expresión] [entero]

Module[{rMax, uMax, omegaGuess}, rMax = FindRmax[l, m, prec];
  [módulo]

  uMax = SetPrecision[1 - 2 / rMax, prec];
    [asigna precisión]

  omegaGuess = QNWKBl, m, n, prec];
  FindRoot[DeltaNumeric[l, m, ω, uMax, Imax, Nmax, prec] == 0,
    [encuentra raíz]

    {ω, omegaGuess}, WorkingPrecision → prec, AccuracyGoal → Floor[prec / 2],
      [precisión operativa] [objetivo de exactitud] [entero inferior]

  PrecisionGoal → Floor[prec / 2], MaxIterations → 200]
    [objetivo de precisión] [entero inferior] [máximo de iteraciones]

In[]:= {time, sol} = AbsoluteTiming[FindQNM1[2, 0, 2, 25, 25, 50]]
  [duración absoluta]

Out[]= {2.90512, {ω → 0.43054416861973394721722579079079882957877931968543 -
  0.50855581055702274054065695543086913367532692171612 I}}}
```