Radial solutions for various harmonics, cribbed from Haberman:

```
V = c2 r^(-(n))
```

$c2\, r^{-n}$

```
D[V, r] / V /. r → R
```

$-\frac{n}{R}$

Can conclude from this that dV/dr + n*V/R

Solutions for each layer of the shell. For the purposes of this worksheet, allow space for up to 20 layers.

```
v = Table[c1[k] * r^(n) + c2[k] * r^(-(n)), {k, 1, 20}];
```

```
dv = D[v, r];
```

```
Bn = (D[v[[1]], r]) * u[1] /. r → R
```

```
(n R^(-1+n) c1[1] - n R^(-1-n) c2[1]) u[1]
```

Define a function to get B.n just inside the inner boundary of the ABC region; this is called as part of the a minimization that computes permeability.

```
GetBn[enn_, dee_, seed_] := Module[{qsub, csub, eqs, m, k, intf, vars},
  m = Length[seed];
  qsub = Table[u[k] → seed[[k]], {k, 1, m}];
  qsub = Join[qsub, {R → 1, n → enn, d → dee}];
  eqs = {(v[[1]] /. r → R /. qsub) == 1, (v[[m]] /. r → R + m * d /. qsub) == 0};
  (* edges of boundary region *)
  For[k = 1, k < m, k++, (* interface conditions between layers in the boundary region *)
   intf = {(u[k] * dv[[k]] /. r → R + k * d /. qsub) ==
      (u[k + 1] * dv[[k + 1]] /. r → R + k * d /. qsub),
     (v[[k]] /. r → R + k * d /. qsub) == (v[[k + 1]] /. r → R + k * d /. qsub)};
   eqs = Join[eqs, intf];
  ];
  vars = Join[Table[c1[k], {k, 1, m}], Table[c2[k], {k, 1, m}]];
  csub = Solve[eqs, vars][[1]];
  Bn /. qsub /. csub
 ]
```

Now, figure out the permeabilities that satisfy the first m asymptotic boundary conditions. Do this via an error minimization. Note that working precision, accuracy, etc., have to be crazy, becuase the calculation isn't well conditioned.

```
GetPerm[een_, guess_] := Module[{ee, uu, m, U, seed, Q, k, ic},
  m = Length[guess];
  ee = Rationalize[een];
  seed = Table[U[k], {k, 1, m}];
  ic = Table[{U[k], guess[[k]]}, {k, 1, m}];
  uu = FindMinimum[Total[Table[(GetBn[k, ee, seed] + k)^2, {k, 1, m}]], ic,
    WorkingPrecision → 10000,
    MaxIterations → 100000, PrecisionGoal → 1000, AccuracyGoal → 1000];
  N[{uu[[1]], seed /. uu[[2]]}]
 ]
```

Test to see if it works. Compute the permeabilities for $\delta = 1$ for a 3 rd order BC, assuming all ones as the initial condition.

```
GetPerm[1, {1, 1, 1}]
```

```
{5.379907774986375 × 10^-2900, {0.998001, 1.26429, 0.209715}}
```

```
pp = {0.001, 0.0025, 0.005, 0.01, 0.025, 0.05, 0.1}
```

```
{0.001, 0.0025, 0.005, 0.01, 0.025, 0.05, 0.1}
```

```
Off[FindMinimum::"precw"];
Off[FindMinimum::"sszero"];
Off[FindMinimum::"fmgz"];
```

Churn through each order, starting at $\delta = 1$ and reducing δ in small steps until $\delta = 0.001$

```
MaxOrder = 9;
iclist = {};
ttt = {};
For[nn = 1, nn ≤ MaxOrder, nn++,
  (* NewGuess=Table[1,{k,1,nn}]; *)
 NewGuess = Join[{1}, iclist];
 tt = {};
 For[k = Length[pp], k > 0, k--,
  NewGuess = GetPerm[pp[[k]], NewGuess][[2]];
  tt = Append[tt, {pp[[k]], NewGuess}];
  Print[pp[[k]], "  ", NewGuess];
  If[k == Length[pp], iclist = NewGuess];
 ];
 ttt = Append[ttt, tt];
]
```

```
For[k = 1, k ≤ MaxOrder, k++, Print[MatrixForm[ttt[[k]]]]]
```

```
( 0.1     {0.0950226}  )
( 0.05    {0.0487515}  )
( 0.025   {0.0246876}  )
( 0.01     {0.00995}   )
( 0.005   {0.0049875}  )
( 0.0025 {0.00249688}  )
( 0.001   {0.0009995}  )
```

```
( 0.1     {3.59547, 0.0586908}   )
( 0.05    {6.88167, 0.0311111}   )
( 0.025   {13.5243, 0.0160781}   )
( 0.01   {33.5097, 0.00656906}   )
( 0.005  {66.8382, 0.00330864}   )
( 0.0025 {133.502, 0.00166045}   )
( 0.001  {333.501, 0.000665669}  )
```

```
( 0.1      {0.514842, 6.56045, 0.0484269}       )
( 0.05     {0.284468, 12.7124, 0.0267313}       )
( 0.025     {0.147067, 24.804, 0.0141267}       )
( 0.01    {0.0596308, 60.8614, 0.00585428}      )
( 0.005   {0.0299163, 120.881, 0.00296304}      )
( 0.0025   {0.0149802, 240.89, 0.00149069}      )
( 0.001  {0.00599693, 600.896, 0.000598504}     )
```

```
 0.1      {1.35828, 0.293044, 8.64352, 0.0426822}
 0.05     {2.21349, 0.145215, 15.9343, 0.024375}
 0.025    {4.1329, 0.0730927, 30.2898, 0.0131449}
 0.01    {10.0833, 0.0296111, 73.1875, 0.00552166}
 0.005   {20.0666, 0.0148952, 144.629, 0.00280808}
 0.0025  {40.0583, 0.00747285, 287.493, 0.00141619}
 0.001  {100.053, 0.00299556, 716.069, 0.000569436}

 0.1      {0.885169, 2.06673, 0.208773, 10.2303, 0.0386362}
 0.05     {0.622287, 4.24886, 0.109025, 18.1433, 0.0227336}
 0.025    {0.354123, 8.67674, 0.0567837, 33.766, 0.0124932}
 0.01     {0.14815, 21.6586, 0.0234358, 80.473, 0.00531697}
 0.005   {0.0746734, 43.1325, 0.0118546, 158.264, 0.00271674}
 0.0025  {0.0374357, 86.0127, 0.00596307, 313.827, 0.00137345}
 0.001  {0.0149914, 214.598, 0.00239404, 780.497, 0.000553067}

 0.1      {1.04258, 0.690526, 2.93883, 0.168805, 11.5866, 0.0354917}
 0.05     {1.30109, 0.353276, 5.96294, 0.0927212, 19.903, 0.0214483}
 0.025    {2.1005, 0.168224, 11.6696, 0.0494806, 36.3371, 0.0119972}
 0.01    {4.85555, 0.0662276, 28.4193, 0.0207194, 85.471, 0.00517088}
 0.005    {9.5826, 0.0331552, 56.2237, 0.010532, 167.304, 0.00265437}
 0.0025 {19.0889, 0.0166132, 111.792, 0.00531093, 330.947, 0.00134515}
 0.001   {47.65, 0.00665722, 278.468, 0.00213539, 821.866, 0.00054247}

 0.1        {0.98778, 1.17266, 0.515654, 3.7876, 0.145127, 12.8231, 0.0329184}
 0.05     {0.874096, 2.01513, 0.242861, 7.23528, 0.0828601, 21.4233, 0.0203775}
 0.025     {0.59794, 4.27556, 0.120239, 13.7234, 0.0451294, 38.4218, 0.0115894}
 0.01    {0.271576, 11.1141, 0.0489475, 32.885, 0.0191523, 89.2377, 0.00505672}
 0.005   {0.138748, 22.3046, 0.0247125, 64.7266, 0.00978172, 173.869, 0.00260766}
 0.0025   {0.0697988, 44.5687, 0.012425, 128.375, 0.0049446, 343.107, 0.00132464}
 0.001  {0.0279787, 111.261, 0.0049877, 319.291, 0.00199105, 850.805, 0.000534985}

 0.1        {1.00316, 0.943838, 1.4155, 0.395672, 4.54967, 0.12888, 13.9897, 0.0307458}
 0.05      {1.06523, 0.64826, 2.94109, 0.193611, 8.25173, 0.0759358, 22.8006, 0.0194524}
 0.025      {1.40958, 0.31387, 6.19958, 0.100001, 15.2894, 0.0421274, 40.215, 0.0112381}
 0.01    {2.91074, 0.117705, 15.4404, 0.0415096, 36.1271, 0.0181083, 92.2656, 0.00496203}
 0.005   {5.62934, 0.0582611, 30.6375, 0.0210732, 70.7654, 0.00929166, 178.949, 0.00257039}
 0.0025    {11.155, 0.0291036, 60.9628, 0.010623, 140.008, 0.0047082, 352.292, 0.00130881}
 0.001  {9.65207, 0.0223912, 105.624, 0.00533832, 300.367, 0.00208798, 825.192, 0.000542153}

 0.1        {0.999291, 1.01807, 0.858296, 1.76775, 0.319548, 5.23532, 0.116696, 15.1123, 0.0288728}
 0.05       {0.973102, 1.27569, 0.462182, 3.80816, 0.16632, 9.12115, 0.0706385, 24.0857, 0.0186345}
 0.025     {0.802638, 2.44969, 0.210289, 7.61892, 0.0884797, 16.5671, 0.0398628, 41.8196, 0.0109261
 0.01     {0.419912, 6.67518, 0.0827166, 18.5036, 0.0372925, 38.6364, 0.0173471, 94.8156, 0.0048802
 0.005   {0.220734, 13.6061, 0.0415767, 36.4891, 0.0190268, 75.3264, 0.00894186, 183.069, 0.0025392
 0.0025    {0.111888, 27.3074, 0.0208815, 72.4048, 0.00961537, 148.672, 0.0045418, 359.549, 0.001296
 0.001  {0.0449472, 68.2569, 0.00837982, 180.108, 0.00387155, 368.679, 0.00183425, 888.967, 0.000524
```

Now, we can create a function that directly gets the collection of permeabilities for any δ and ABC order :

```
GetPermAlt[dee_, enn_] := Module[{seed},
  seed = Interpolation[ttt[[enn]], InterpolationOrder → 1][dee];
  GetPerm[dee, seed][[2]]
 ]
```

Table of permeabilities needed for 0.1 R stackup case

```
MatrixForm[Table[{1 / (10 k), GetPermAlt[1 / (10 k), k]}, {k, 1, MaxOrder}]]
```

```
( 1/10                                   {0.0950226}
  1/20                               {6.88167, 0.0311111}
  1/30                         {0.194197, 18.7726, 0.0184821}
  1/40                    {4.1329, 0.0730927, 30.2898, 0.0131449}
  1/50               {0.288508, 10.8586, 0.0458772, 41.5572, 0.0101984}
  1/60           {2.99677, 0.11082, 17.2721, 0.0338143, 52.7245, 0.00833053}
  1/70     {0.377373, 7.72003, 0.0694494, 23.3195, 0.0268835, 63.8397, 0.00704076}
  1/80   {2.38424, 0.148376, 12.3865, 0.0515218, 29.1926, 0.0223527, 74.9238, 0.00609673}
  1/90 {0.459734, 5.97251, 0.0918638, 16.7005, 0.0412586, 34.9645, 0.0191478, 85.988, 0.00537587} )
```