Computations of the alternative expansion for self-avoiding walk

The expansion of Chapter 6 with the Analysis of Section 3.3

Robert Fitzner
University of Technology Eindhoven

Abstract

This file performs the computations for the analysis of the lace expansion explained in Chapter 6 of the PhD thesis of the author. If succesful the computations complete the proof of mean-field behavior for the nearest-neighbor self-avoiding walk in a given dimension. All references in this version of the notebook will be to the PhD thesis. The first three sections in the notebook are the same as in the document for the SAW-NoBLE computations.

As input the dimension d and the constance Γ_1,Γ_2,Γ_3,c_2,c_4 are expected. After choosing these quantities we select the menu item Evaluate-> Evaluate Notebook. In a table at the end of this document we will then be shown whether the bootstrap and thereby the analysis is succesful.

We first compute bounds on the simple random walk two-point function (Section 5.2). Then we compute bound on the two-point function and repulsive diagrams (Section 5.1.2). We use these bounds on diagrams to compute the bound on the coefficients stated and proven in Section 6.4-6.5 and compute the bounds used for the Analysis in Section 3.3 and 6.6.

14.05.2013

Input

In which dimension should we perform the computation

```
d = 7;
```

For the bootstrap we assume that $f_i(z) \leq \Gamma_i$ with Γ_i gives as follows

```
Gamma1 = 1.021; Gamma2 = 1.032; Gamma3 = 1.1;
```

We define bootstrap function $f_3$with the following constants c_2 and c_4.

```
c2 = 0.5; c4 = 4.13;
```

As the nominator $\Phi_z$of the G_z in line (7.6.1) is trivial $c_1$and $c_3$are not need. (See also the analysis of Section 4.3.5)

(*d=7;Gamma1=1.021;Gamma2=1.032;Gamma3=1.1;c2=0.5;c4=4.13;*)

Simple Random Walk integral

Overview

We compute the two-point function of the simple random walk,

$$I_{n,m}(x) = \int_{[-\pi,\pi]} e^{\mathrm{i}k\,x} \frac{\hat{D}^m(k)}{\left(1-\hat{D}(k)\right)^n} \frac{d^d k}{(2\pi)^d}$$

see Section 5.2, using that

$$I_{n,m}(x) = I_{n,(m-1)}(x) - I_{(n-1),(m-1)}(x)$$

$$I_{n,0}(x) = \frac{1}{(n-1)!} \int_0^\infty t^{n-1} \prod_{i=1}^d \int_{-\pi}^{\pi} e^{-t/d(1-\cos(k_i))} e^{\mathrm{i}k_i\,x_i} \frac{\mathrm{dk}}{2\pi} = \frac{1}{(n-1)!} \int_0^\infty t^{n-1} \prod_{i=1}^d F(t,\, d,\, |x_i|)$$

where F(t,d,n) is the modified Besselfunction. We implement the Besselfunciton and a function to compute $I_{n,0}(0)$.

```
F[t_, d_, N_] := e^(-t/d) BesselI[N, t/d];
NInt[n_, d_, T_] := 1 / ((n - 1) !) * NIntegrate[t^(n - 1) * (F[t, d, 0])^d, {t, 0, T}]
```

Computation

Then, we define the number of n-step SRW loop as given in Section 5.2.2, see (5.2.6)-(5.2.10)

```
s2 = N[2 d];
s4 = N[(d * 4! / (2 × 2) + d (d - 1) / 2 * 4!)];
s6 = N[ (d * 6! / (3! 3!) + d * (d - 1) * 6! / (2 × 2) + d (d - 1) (d - 2) / 3! * 6!)];
s8 =
  N[(d * 8! / (4! 4!) + d * (d - 1) * (8! / (3! 3!) + 8! / 2^5) + d (d - 1) (d - 2) / 2 * 8! / (2 × 2) +
      d (d - 1) (d - 2) (d - 3) / 4! * 8!)];
```

and compute the number $I_{n,m}(0)$ for n=0,1,2 and m=1,...,10 and save them in an two-dimensional array:

```
TableTwoPointValues = {{n m, 0, 1, 2}, {0, 1, NInt[1, d, ∞], NInt[2, d, ∞]},
   {1, 0, 0, 0}, {2, s2 / (2 d)^2, 0, 0}, {3, 0, 0, 0}, {4, s4 / (2 d)^4, 0, 0},
   {5, 0, 0, 0}, {6, s6 / (2 d)^6, 0, 0}, {7, 0, 0, 0}, {8, s8 / (2 d)^8, 0, 0}, {9, 0, 0, 0},
   {10, -1, 0, 0}};
For[i = 3, i < 13, i++,
 For[j = 3, j < 5, j++,
  TableTwoPointValues[[i, j]] =
    TableTwoPointValues[[i - 1, j]] - TableTwoPointValues[[i - 1, j - 1]];]
]
Clear[i, j]
```

For latter reference we define the constants:

```
I10 = TableTwoPointValues[[2, 3]];
I11 = TableTwoPointValues[[3, 3]];
I12 = TableTwoPointValues[[4, 3]];
I14 = TableTwoPointValues[[6, 3]];
I16 = TableTwoPointValues[[8, 3]];
I18 = TableTwoPointValues[[10, 3]];
I110 = TableTwoPointValues[[12, 3]];
I20 = TableTwoPointValues[[2, 4]];
I21 = TableTwoPointValues[[3, 4]];
I22 = TableTwoPointValues[[4, 4]];
I24 = TableTwoPointValues[[6, 4]];
I26 = TableTwoPointValues[[8, 4]];
I28 = TableTwoPointValues[[10, 4]];
I210 = TableTwoPointValues[[12, 4]];
```

Print out of all computed values of $I_{n,m}(0)$

```
Labeled[Grid[TableTwoPointValues,
   Alignment -> {{Left, Center}, Baseline, {{2, 12}, {2, 3}} -> {"."}},
   Frame -> True, Dividers -> {{2 -> True, -1 -> True}, {2 -> True}},
   Spacings -> {1.5, {1.5, 1, {0.5}}}, ItemStyle -> {1 -> Bold, 1 -> Bold},
   Background -> {Automatic, Automatic, {{2, 12}, {2, 4}} -> GrayLevel[0.9]}],
  Style["Value of the SRW two-point function in dimension " Text[d] , Bold],
  Top] // Text
```

Value of the SRW two−point function in dimension 7

m n	**0**	**1**	**2**
0	1	1.09391	1.36679
1	0	0.0939063	0.27288
2	0.0714286	0.0939063	0.178974
3	0	0.0224777	0.0850672
4	0.0142128	0.0224777	0.0625895
5	0	0.00826492	0.0401117
6	0.00436946	0.00826492	0.0318468
7	0	0.00389546	0.0235819
8	0.00174148	0.00389546	0.0196864
9	0	0.00215398	0.015791
10	−1	0.00215398	0.013637

Bound on the two-point function and on repulsive diagrams

Definition of Constants

We consider two setting s: we use s=i for bound on $z = z_I$ and s=o for bound on $z\epsilon\,(z_I,\ z_c)$. In Section 5.1 we argued that the bounds have a similar for. As given in (5.1.13) and (5.1.14) we define

```
z[i] = 1/(2 d - 1);
z[o] = Gamma1/(2 d - 1);
VarGamma2[i] = 2 (d - 1)/(2 d - 1);(* G_{1/(2d-1)}(x) >=B_{1/(2d-1)}(x) = (2d-2)/(2d-1) C_{1/2d}(x)*)
VarGamma2[o] = 2 (d - 1)/(2 d - 1) Gamma2;
```

Further, we define the constants given in Section 5.1.3.

```
c2ik = 2; (*c_2(e_1+e_2)*)
c4ik = 4 + 6 (2 d - 4);(*c_4(e_1+e_2)*)
c6ik = 16 + 84 (2 d - 4) + 36 (2 d - 4) (2 d - 6);(*c_4(e_1+e_2)*)

c3i = (2 d - 2);(*c_3(e_1)*)
c5i = (3 (2 d - 2 ) + 4 (2 d - 2) (2 d - 4) );(*c_5(e_1)*)
c7i = (14 (2 d - 2 ) + 62 (2 d - 2) (2 d - 4) + 27 (2 d - 2) (2 d - 4) (2 d - 6) )(*c_7(e_1)*)
```

33 528

Bounds on two-point function

We compute the bounds as explained in Section 5.1.2.We begin by computing $G_{n,z}(e_1)$ as in (5.1.22)-(5.1.24)

```
Do[
 Bound[G7i, s] = c7i z[s]^7 + (2 d * z[s])^9 * VarGamma2[s] * I110; (* G7,z(e1)*)
 Bound[G5i, s] = c5i z[s]^5 + Bound[G7i, s]; (* G5,z(e1)*)
 Bound[G3i, s] = c3i z[s]^3 + Bound[G5i, s]; (* G3,z(e1)*)
 Bound[G1i, s] = z[s] + Bound[G3i, s];(* G1,z(e1)*)
 , {s, {i, o}}]
```

Then, we compute $G_{n,z}(e_1 + e_2)$ and $G_{4,z}(2\, e_2)$, see (5.1.25)-(5.1.26) and (5.4.1):

```
Do[
 Bound[G8ik, s] = d/(d - 1) (2 d * z[s])^8 VarGamma2[s] I110; (*G8(e1+e2)*)
 Bound[G6ik, s] = c6ik z[s]^6 + VarGamma2[s] I18; (*G6(e1+e2)*)
 Bound[G4ik, s] = Bound[G6ik, s] + (c4ik - 2 (2 d - 3) ) z[s]^4; (*G4^1(e1+e2)*)
 Bound[G2ik, s] = Bound[G4ik, s] + (c2ik - 1) z[s]^2; (*G2^1(e1+e2)*)
 Bound[G4ii, s] = (2 d + 2) z[s]^4 + Bound[G6, s];(* Bound for sup_x G4^1(2 e1)*)
 , {s, {i, o}}]
```

We compute the supreme of the two-point function as given in (5.1.28)-(5.1.31):

```
Do[
 Bound[G6, s] = Max[c6ik z[s]^6, c7i z[s]^7] + (2 d * z[s])^8 VarGamma2[s] I18;
 (* Bound for sup_x G6(x)*)
 Bound[G4, s] = Max[c4ik z[s]^4, c5i z[s]^5] + Bound[G6, s]; (* Bound for sup_x G4(x)*)
 Bound[G2, s] = Max[c2ik z[s]^2, c3i z[s]^3] + Bound[G4, s]; (* Bound for sup_x G2(x)*)
 Bound[G1, s] = Max[Bound[G1i, s], Bound[G2, s]];(* Bound for sup_x G1(x)*)
 , {s, {i, o}}]
```

Bounds on repulsive bubbles

As next we bound the closed diagrams, see (5.1.37). Further, we use that $B_{\underline{1},3}(0) = 2\, d\, z\, G_{3,z}(e_1)$ and $B_{1,2}(0) = B_{\underline{1},3}(0) + B_{2,2}(0)$

```
Do[
 RepBubble[Step, G3, s] = 2 d z[s] Bound[G3i, s] ;
 RepBubble[G2, G2, s] = 2 d c3i z[s]^4 + 2 d 3 c5i z[s]^6 + 2 d 5 c7i z[s]^8 +
   (2 d z[s])^10 (4 VarGamma2[s] I110 + VarGamma2[s]^2 I210);
 RepBubble[G1, G2, s] = RepBubble[Step, G3, s] + RepBubble[G2, G2, s];
 , {s, {i, o}}]
```

We bound the open diagram $B_{2,0}(x)$ as shown in (5.1.38) and see that we can use the same bound for $B_{1,1}(x)$ and that $B_{0,1}(x) = B_{\underline{0},1}(x) + B_{1,1}(x)$:

```
Do[
 OpenRepBubble[G2, G0, s] = Max[c2ik z[o]^2 + 3 c4ik z[o]^4, 2 c3i z[o]^3 + 4 c5i z[o]^5] +
   4 (2 d z[o])^6 VarGamma2[o] I16 + (2 d z[o])^6 VarGamma2[o]^2 I26;
 OpenRepBubble[G1, G1, s] = OpenRepBubble[G2, G0, s];
 OpenRepBubble[G0, G1, s] = Bound[G1, s] + OpenRepBubble[G1, G1, s];
 , {s, {i, o}}]
```

Bounds on weighted diagrams

We bound $\sup_x G_z(x)[1 - \cos(k\, x)]$ as how in (5.1.19) and (5.1.20):

```
Bound[DeltaG, o] = (2 c2 + c4) I20 Gamma3 ;
Bound[DeltaG, i] = (2 d - 2)/(2 d - 1) (I20 + 4 I20) ;
```

Further, we bound $\sum_x G_{2,z}(x)^2\, G_z(x)[1 - \cos(k\, x)]$ as explained in Section 5.4.1 and given (5.4.7). For $z = z_I$ $(s = i)$ we use (5.1.20) and (5.4.7) with $c_1 = 0,\ c_3 = 0,\ c_2 = 0.5$:

```
Bound[G2CubedAndOneWeight, o] :=
  (2 d z[o])^4 VarGamma2[o]^2 Gamma3 (2 c2 (I12 + I22) (I24 - I12^2) + 2 c2 I20 I12^2);
Bound[G2CubedAndOneWeight, i] :=
  (2 d z[i])^4 (2 (d - 1) / (2 d - 1))^3 ((I12 + I22) (I24 - I12^2) + I20 I12^2);
```

Bound on the coefficient Φ_z^ι

Bound for N=1

We implement the bounds given in Section 6.4.1:

```
Do[
 Bound[Phi, 1, ii, s] = z[s]^2 Bound[G4ii, s];
 Bound[Phi, 1, iin, s] = 0;
 Bound[Phi, 1, ik, s] = z[s]^2 Bound[G2ik, s];
 Bound[Phi, 1, sumi, s] = z[s] Bound[G3i, s];
 Bound[Phi, 1, Delta, s] = 0;
 , {s, {i, o}}]
```

Bound for N=2

Then the bounds given in Section 6.4.2:

```
Do[
 bii = z[s]^2 Bound[G3i, s] + z[s]^2 Bound[G4ii, s]^2 Bound[G1i, s] +
   (2 d - 2) z[s]^2 Bound[G2ik, s]^2 Bound[G1i, s];
 biin = 2 z[s]^2 Bound[G3i, s] * Bound[G4ii, s] +
   (2 d - 2) z[s]^2 Bound[G2ik, s]^2 Bound[G1i, s];
 bik = 2 z[s]^2 Bound[G2ik, s] * Bound[G1i, s] (Bound[G4ii, s] + Bound[G3i, s]) +
   (2 d - 4) z[s]^2 Bound[G1i, s] Bound[G2ik, s]^2;
 bisumi = z[s] Bound[G3i, s] Bound[G1i, s] + z[s] Bound[G4ii, s] Bound[G1i, s]^2 +
   (2 d - 2) z[s] Bound[G2ik, s] (z[s] + Bound[G1i, s]) Bound[G3i, s];
 Bound [Phi, 2, ii, s] = bii + z[s]^2 Bound[G1, s] RepBubble[G2, G2, s] / (2 d z[s]);
 Bound [Phi, 2, iin, s] = biin + z[s]^2 Bound[G1, s] RepBubble[G2, G2, s] / (2 d z[s]);
 Bound [Phi, 2, ik, s] = bik + z[s]^2 Bound[G1, s] RepBubble[G2, G2, s] / (2 d z[s]);
 Bound [Phi, 2, sumi, s] = bisumi + 2 d z[s]^2 Bound[G1, s] RepBubble[G2, G2, s] / (2 d z[s]);
 Bound [Phi, 2, Delta, s] = 2 d bisumi + Bound[G2CubedAndOneWeight, s];
 Clear[bii, biin, bik, bisumi];
 , {s, {i, o}}]
```

Bound for N≥3

The implementation of the bounds stated in Section 6.4.3:

Declaration of variables

We implement the vectors and matricies as given in (6.4.7)-(6.4.10):

```
Do[
 Vector[vi, s] = {z[s]/(2 d) Bound[G3i, s]^2 + z[s] Bound[G1, s] RepBubble[G2, G2, s]/(2 d z[s]),
   z[s] Max[1, Bound[G1, s]] (2 Bound[G3i, s] + RepBubble[G2, G2, s]/(2 d z[s]))};
 MatrixB[s] = {{Bound[G3i, s], Max[z[s], Bound[G2, s]]},
   {Bound[G3i, s] + RepBubble[G2, G2, s]/(2 d z[s]), OpenRepBubble[G2, G0, s]}};
 Vector[w, s] = { 2 d z[s] Bound[G3i, s], RepBubble[G1, G2, s]};
 , {s, {i, o}}]
```

To compute the sum of the bound we compute the eigensystem of B and a decomposition of *w*and (1,1) as explained in Section 5.3:

```
Do[
 EigensystemB[s] = Eigensystem[MatrixB[s]];
 Do[
  VectorB[v, j, s] = EigensystemB[s][[2, j]] *
    (Inverse[Transpose[ EigensystemB[s][[2]] ]].Vector[w, s])[[j]];
  EigenValueB[j, s] = EigensystemB[s][[1, j]];
  , {j, {1, 2}}];
 , {s, {i, o}}]
```

Bound on the absolute value (k=0)

```
Do[
 Bound[Phi, OddTail, sumi, s] = Sum[Vector[vi, s].VectorB[v, j, s]/(1 - EigenValueB[j, s]^2), {j, 1, 2}];
 Bound[Phi, EvenTail, sumi, s] =
  Sum[Vector[vi, s].VectorB[v, j, s] EigenValueB[j, s]/(1 - EigenValueB[j, s]^2), {j, 1, 2}];
 Do[
  Bound[Phi, EvenTail, t, s] = Bound[Phi, EvenTail, sumi, s] / (2 d);
  Bound[Phi, OddTail, t, s] = Bound[Phi, OddTail, sumi, s] / (2 d);
  , {t, {ii, iin, ik}}];
 , {s, {i, o}}]
```

Bounds on Difference

```
Do[
 Bound[Phi, OddTail, Delta, s] =
  Bound[DeltaG, s] (Sum[{1, 1}.VectorB[v, j, s] / (1 - EigenValueB[j, s]^2), {j, 1, 2}])
   (Sum[2 * {1, 1}.VectorB[v, j, s] / (1 - EigenValueB[j, s]^2)^2 - {1, 1}.VectorB[v, j, s] / (1 - EigenValueB[j, s]^2), {j, 1, 2}]);
 Bound[Phi, EvenTail, Delta, s] =
  Bound[DeltaG, s] (Sum[{1, 1}.VectorB[v, j, s] EigenValueB[j, s] / (1 - EigenValueB[j, s]^2), {j, 1, 2}])
    (Sum[{1, 1}.VectorB[v, j, s] / (1 - EigenValueB[j, s]^2)^2, {j, 1, 2}]) +
   Bound[DeltaG, s] (Sum[{1, 1}.VectorB[v, j, s] / (1 - EigenValueB[j, s]^2), {j, 1, 2}])
    (Sum[{1, 1}.VectorB[v, j, s] EigenValueB[j, s] / (1 - EigenValueB[j, s]^2)^2, {j, 1, 2}]) +
   Sum[VectorB[v, j, s] EigenValueB[j, s] / (1 - EigenValueB[j, s]^2)^2 + VectorB[v, j, s] EigenValueB[j, s] / (1 - EigenValueB[j, s]^2),
     {j, 1, 2}].
    {z[s] Bound[G3i, s] + Bound[DeltaG, s] * (Bound[G3i, s] + RepBubble[G2, G2, s] / (2 d z[s])),
     Bound[DeltaG, s] * OpenRepBubble[G0, G1, s]};
 , {s, {i, o}}]
```

Sum over N

We combine the bound we computed in the preceding sections and compute the following bound on the absolute value of Φ_z

```
Do[
 Do[
  Bound[Phi, Even, t, s] = Bound[Phi, EvenTail, t, s] + Bound[Phi, 2, t, s];
  Bound[Phi, Odd, t, s] = Bound[Phi, OddTail, t, s] + Bound[Phi, 1, t, s];
  Bound[Phi, abs, t, s] = Bound[Phi, Even, t, s] + Bound[Phi, Odd, t, s];
  , {t, {ii, iin, ik, sumi}}];
 Bound[Phi, abs, max, s] = Max[Bound[Phi, abs, ii, s], Bound[Phi, abs, iin, s],
   Bound[Phi, abs, ik, s]];
 , {s, {i, o}}]
```

For the bound on the difference we compute

```
Do[
 Bound[Phi, abs, Delta, s] =
  Sum[Bound[Phi, r, Delta, s], {r, {1, 2, EvenTail, OddTail}}];
 Bound[Phi, Odd, Delta, s] = Sum[Bound[Phi, r, Delta, s], {r, {1, OddTail}}];
 Bound[Phi, Even, Delta, s] = Sum[Bound[Phi, r, Delta, s], {r, {2, EvenTail}}];
 , {s, {i, o}}]
```

Bound on the coefficient Π_z

Bound on Π_z

We first compute the bound, as shown in Section 6.5.1:

```
Do[
 Bound[Pi, upper, s] = - (2 d z[i]^2)/(1 + z[i]) + (2 d)/(1 + z[s]) (Bound[Phi, Even, sumi, s])/(1 + z[s] - Bound[Phi, Odd, sumi, s]);
 Bound[Pi, lower, s] = - (2 d z[s]^2)/(1 + z[s]) - (2 d)/(1 + z[i]) (Bound[Phi, Odd, sumi, s])/(1 + z[i] - Bound[Phi, Odd, sumi, s]);
 , {s, {i, o}}]
```

Bound on $\sum_{ikx} [1 - \cos(kx)]\, \Pi^{ik}{}_z(x)$

Now we compute bounds: a lower bound for $\sum_{ikx} [1-\cos(kx)]\, \Pi^{ik}{}_z(x)$ and an upper bound for $\sum_{ikx} [1-\cos(kx)]\, \left| \Pi^{ik}{}_z(x) \right|$, see Section 6.5.2 and 6.5.3. We bound these differences by expanding it into eleven terms, see (6.5.7) and (6.5.10)-(6.5.14). We first bound the four explict terms. Thereby we use the label low, to denote that this corresponds to the lower bound for $\sum_{ikx} [1-\cos(kx)]\, \Pi^{ik}{}_z(x)$. Everything that we use to bound $\sum_{ikx} [1-\cos(kx)] \left| \Pi^{ik}{}_z(x) \right| = \Pi^{AV}(0) - \Pi^{AV}(k)$ will carry the label AV.

```
Do[
 Bound[PiPart, Main, low, s] = (2 d z[i]^3)/(1 - z[i]^2); (*(7.5.7)*)
 Bound[PiPart, Main, AV, s] = (2 d z[s]^3)/(1 - z[s]^2); (*(7.5.7)*)
 Bound[PiPart, 1, low, s] = (2 d z[s]^3)/((1 - z[s]^2) (1 + z[s])) (Bound[Phi, Odd, sumi, s])/(1 - Bound[Phi,Odd,sumi,s]/(1+z[i]));
 (*(7.5.15)*)
 Bound[PiPart, 1, AV, s] = (2 d z[s])/((1 - z[s]^2) (1 - z[s])) (Bound[Phi, abs, sumi, s])/(1 - Bound[Phi,abs,sumi,s]/(1-z[s]));
 (*(7.5.15)*)
 Bound[PiPart, 2, low, s] = Bound[PiPart, 1, low, s]; (*(7.5.16)*)
 Bound[PiPart, 2, AV, s] = Bound[PiPart, 1, AV, s]; (*(7.5.16)*)

 Bound[PiPart, 3, low, s] = (z[s]/(1 + z[s]))^2 (Bound[Phi, Odd, Delta, s])/(1 - Bound[Phi,Odd,sumi,s]/(1+z[i])); (*(7.5.17)*)
 Bound[PiPart, 3, AV, s] = (z[s]/(1 - z[s]))^2 (Bound[Phi, abs, Delta, s])/(1 - Bound[Phi,abs,sumi,s]/(1-z[s]));(*(7.5.17)*)
 , {s, {i, o}}]
```

The other terms are bounded using matrix norms. We by computing these norms. See (6.5.25)-(6.5.37)

```
Do[
 Bound[EucleadOfVZero, s] = 1/(1 - z[s]) Sqrt[2 d];
 Bound[EucleadOfVZeroMinusVk, s] = z[s]/(1 - z[s]^2) Sqrt[4 d];
 Bound[EucleadBZeroOneV, low, s] = Sqrt[2 d] 1/(1 - Bound[Phi,Odd,sumi,s]/(1+z[s]));
 Bound[EucleadBZeroOneV, AV, s] = Sqrt[2 d] 1/(1 - Bound[Phi,abs,sumi,s]/(1-z[s]));
 Bound[EucleadPhiZeroOneV, low, s] =
  Sqrt[2 d] Max[Bound[Phi, Even, sumi, s], Bound[Phi, Odd, sumi, s]];
 Bound[EucleadPhiZeroOneV, AV, s] = Sqrt[2 d] Bound[Phi, abs, sumi, s];
 Bound[OPPhik, s] = Bound[Phi, abs, sumi, s];

 Bound[HSPhiZeroMinusPhik, s] =
  Sqrt[2 Bound[Phi, abs, max, s] Bound[Phi, abs, Delta, s]];
 Bound[HSPhiZeroMinusPhikDk, s] =
  Bound[HSPhiZeroMinusPhik, s] + Sqrt[4 d] Bound[Phi, abs, max, s];

 Bound[HSAzeroMinusAk, s] = 1/(1 - z[s]^2) Bound[HSPhiZeroMinusPhik, s] +
   z[s]/(1 - z[s]^2) Bound[HSPhiZeroMinusPhikDk, s];
 , {s, {i, o}}]
```

To compute the remaining norms we need to compute operator norm of A_k=Φ(k)Υ(k). We compute the eigenvalues of A_o to $\frac{1}{1+z}\,\Phi^{ii}$, $\frac{-1}{1+z}\left(\Phi^{ii}-\Phi^{i-i}\right)$, $\frac{-1}{1+z}\left(\Phi^{ii}+\Phi^{ii}-2\,\Phi^{ik}\right)$. The biggest of these values equals the operator norm of A_o. To bound the norm of $A_k$for generel k we bound $\Phi_o$entry-wise by C, (6.5.4) and then bound the norm of the created matrix. As all entries of this matrix are non-negative it is clear biggest eigenvalue is given by the row sum.

```
Do[
 EigenvalueAZero[1, low, s] =
   1/(1 + z[s]) Max[Bound[Phi, Even, sumi, s], Bound[Phi, Odd, sumi, s]];
 EigenvalueAZero[2, low, s] =
   1/(1 + z[s]) Max[Bound[Phi, Odd, ii, s] + Bound[Phi, Odd, iin, s],
     Bound[Phi, Even, ii, s] + Bound[Phi, Even, iin, s]];
 EigenvalueAZero[3, low, s] =
   1/(1 + z[s]) Max[Bound[Phi, Odd, ii, s] + Bound[Phi, Odd, iin, s] +
      2 Bound[Phi, Even, ik, s], Bound[Phi, Even, ii, s] + Bound[Phi, Even, iin, s] +
      2 Bound[Phi, Odd, ik, s]];

 Bound[OPAZero, low, s] = Max[EigenvalueAZero[1, low, s], EigenvalueAZero[2, low, s],
    EigenvalueAZero[3, low, s]];
 Bound[OPAZero, AV, s] = 1/(1 - z[s]) Bound[Phi, abs, sumi, s];
 Bound[OPAk, low, s] = 1/(1 - z[s]) Bound[Phi, abs, sumi, s];
 Bound[OPAk, AV, s] = 1/(1 - z[s]) Bound[Phi, abs, sumi, s];
 , {s, {i, o}}]
```

We continue by computing the remaining bound (6.5.27)

```
Do[
 Do[
   Bound[OPBZero, t, s] = 1/(1 - Bound[OPAZero, t, s]);
   Bound[OPBk, t, s] = 1/(1 - Bound[OPAk, t, s]);
   Bound[HSBZeroMinusBk, t, s] =
    Bound[OPBZero, t, s] Bound[HSAzeroMinusAk, s] Bound[OPBk, t, s]
   , {t, {low, AV}}];
 , {s, {i, o}}]
```

Then we bound the remaining seven terms of (6.5.10)-(6.5.14):

```
Do[
 Do[
   Bound[PiPart, 4, t, s] = Bound[EucleadBZeroOneV, t, s] Bound[HSAzeroMinusAk, s]
     Bound[OPBk, t, s] Bound[EucleadPhiZeroOneV, t, s];
   Bound[PiPart, 5, t, s] = Bound[EucleadOfVZeroMinusVk, s]
     Bound[HSBZeroMinusBk, t, s] Bound[EucleadPhiZeroOneV, t, s];
   Bound[PiPart, 6, t, s] = Bound[EucleadOfVZero, s] Bound[HSBZeroMinusBk, t, s]
     Bound[HSPhiZeroMinusPhik, s] Bound[EucleadOfVZero, s];
   Bound[PiPart, 7, t, s] = Bound[EucleadBZeroOneV, t, s] Bound[HSPhiZeroMinusPhik, s]
     Bound[EucleadOfVZeroMinusVk, s];
   Bound[PiPart, 8, t, s] = Bound[EucleadOfVZeroMinusVk, s] Bound[OPBk, t, s]
     Bound[HSPhiZeroMinusPhik, s] Bound[EucleadOfVZero, s];
   Bound[PiPart, 9, t, s] = Bound[EucleadOfVZero, s] Bound[HSBZeroMinusBk, t, s]
     Bound[OPPhik, s] Bound[EucleadOfVZeroMinusVk, s];
   Bound[PiPart, 10, t, s] = Bound[EucleadOfVZeroMinusVk, s] Bound[OPBk, t, s]
     Bound[OPPhik, s] Bound[EucleadOfVZeroMinusVk, s]
   , {t, {low, AV}}];
 , {s, {i, o}}]
```

Finally, we sum these terms to create the required bounds

```
Do[
 Bound[Pi, Delta, low, s] =
  Bound[PiPart, Main, low, s] - Sum[Bound[PiPart, j, low, s], {j, 1, 10}];
 Bound[Pi, Delta, AV, s] =
  Bound[PiPart, Main, AV, s] + Sum[Bound[PiPart, j, AV, s], {j, 1, 10}];
 , {s, {i, o}}]
Clear[j, l];
```

Computation of constants of Proposition 3.3.1

In this section we perform the computations of Section 3.4. to obtain the constances stated in Proposition 3.3.1

$$\begin{aligned} \sum_x |F(x)| &\leq K_F = \text{Bound[KF]} \\ \sum_x F(x)[1-\cos(k\,x)] &\geq K_{\text{Lower}}\left[1-\hat{D}(k)\right] \\ \sum_x \left|F(x)\right|[1-\cos(k\,x)] &\leq K_{\Delta\text{F}}\left[1-\hat{D}(k)\right] \end{aligned} \tag{1}$$

and recall that Φ is trivial for this expansion

$$\begin{aligned} \underline{K}_\Phi = \hat{\Phi}(0) = \overline{K}_\Phi = \underline{K}_{|\Phi|} = \sum_x |\Phi(x)| = \overline{K}_{|\Phi|} = 1 \\ \sum_{x\neq 0} |\Phi_z(x)| \leq K_{|\Phi|} = 0 \\ \sum_x \left|\Phi_z(x)\right|[1-\cos(k\,x)] \leq K_{\Delta\Phi}\left[1-\hat{D}(k)\right] = 0 \end{aligned} \tag{2}$$

Computation of the constants

Bound on the values as given in Section 3.4.2, thereby we recall that this Phi was trivail:

```
Do[
 Bound[KF, s] = 2 d z[s] + Bound[Pi, upper, s];
 Bound[KDeltaFLower, s] = 1 / (2 d z[i] + Bound[Pi, Delta, low, s]);
 Bound[KDeltaF, s] = 2 d z[o] + Bound[Pi, Delta, AV, s];
 , {s, {i, o}}]
```

Check of the sufficient condition

Now we can compute whether P(γ,Γ,z) is satisfied, see Definition 3.3.2.

```
LEf1Bound[i] = 1;
LEf1Bound[o] = (2 d - 1)/(2 d) (1 - Bound[Pi, lower, o]);

LEf2Bound[i] = (2 d - 1)/(2 d - 2) Bound[KDeltaFLower, i];
LEf2Bound[o] = (2 d - 1)/(2 d - 2) Bound[KDeltaFLower, o];
```

Can compute the bound on f_3:

```
Do[
 LEf3Bound[Part2, s] =
  If[c2 ≤ 0, 1000, 1/(2 c2) Bound[KDeltaF, s] Bound[KDeltaFLower, s]^2];
 LEf3Bound[Part4, s] =
  If[c4 ≤ 0, 1000, 4/c4 Bound[KDeltaFLower, s]^3 Bound[KDeltaF, s]^2];
 LEf3Bound[s] = Max[LEf3Bound[Part2, s], LEf3Bound[Part4, s]];
 , {s, {i, o}}]

Do[
 Succes[f1, s] = LEf1Bound[s] < Gamma1;
 Succes[f2, s] = LEf2Bound[s] < Gamma2;
 Succes[f3, s] = LEf3Bound[s] < Gamma3;
 Succes[s] = Succes[f1, s] && Succes[f2, s] && Succes[f3, s];
 , {s, {i, o}}]
Succes[overall] = Succes[i] && Succes[o];
```

Outputs

The overall result

The statement that the bootstrap was succesful is

```
Succes[overall]
```

```
True
```

If this succedes than the analysis of Section 3.3 states mean-field behavoior for SAW.

Assuming the bootstrap was succesful we prove the following bounds: The infrared bound in (3.3.12) and (3.3.13) hold with

```
(2 d - 2)/(2 d - 1) Gamma2(* ≥ G_z(k) [1-D̂(k)] *)
Bound[KDeltaFLower, o]
(* Nominator in (4.3.13) *)
```

```
0.952615
```

```
0.952375
```

Further we have proven that z_c is smaller than

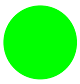
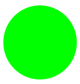
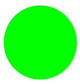
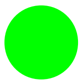
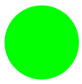
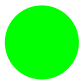
$$\frac{1}{2\,d-1}\,\text{Gamma1}$$

0.0785385

The improvement of bounds

```
bubbles = {Graphics[{Green, Disk[{0, 0}, 0.8]}, ImageSize -> 40],
   Graphics[{Red, Disk[{0, 0}, 0.8]}, ImageSize -> 40]};
MethodeFourTablePart3 = {{Quantity, F1, F2, F3Part2, F3Part4, F2 - init, F3 - init},
  {"To beat", Gamma1, Gamma2, Gamma3, Gamma3, Gamma2, Gamma3},
  {Computed, LEf1Bound[o], LEf2Bound[o], LEf3Bound[Part2, o], LEf3Bound[Part4, o],
   LEf2Bound[i], LEf3Bound[i]},
  {check, If[Succes[f1, o], bubbles[[1]], bubbles[[2]]],
   If[Succes[f2, o], bubbles[[1]], bubbles[[2]]],
   If[LEf3Bound[Part2, o] < Gamma3, bubbles[[1]], bubbles[[2]]],
   If[LEf3Bound[Part4, o] < Gamma3, bubbles[[1]], bubbles[[2]]],
   If[Succes[f2, i], bubbles[[1]], bubbles[[2]]],
   If[LEf3Bound[i] < Gamma3, bubbles[[1]], bubbles[[2]]]},
  {Required c_i, , , LEf3Bound[Part2, o] * c2 / Gamma3, LEf3Bound[Part4, o] * c4 / Gamma3},
  {given c_i , , , c2, c4}};
Labeled[Grid[MethodeFourTablePart3, Alignment → {Center}, Frame → True,
   Dividers → {{2 → True, -1 → True}, {2 → True}}, ItemStyle → {1 → Bold, 1 → Bold},
   Background → {{None}, {GrayLevel[0.9]}, {None}}],
  Style["Bound on coefficients in Dimension " Text[d], Bold], Top] // Text
```

Bound on coefficients in Dimension 7

Quantity	F1	F2	F3Part2	F3Part4	F2 – init	F3 – init
To beat	1.021	1.032	1.1	1.1	1.032	1.1
Computed	1.02076	1.03174	1.03973	1.09936	1.02058	1.03882
check						
Required c_i			0.472604	4.12761		
given c_i			0.5	4.13		

The table above shows whether the improvement of bound for the function f_i was succesfull. As the bound on f_3 is the hardest to improve we show both parts. To be able to choose the value of c_i correctly me mark the required values in the table.