Part VI

Appendix
A

Dimensional Reduction of 10-dimensional Super-Yang-Mills Theory

We will review the dimensional reduction of super Yang-Mills theory following [90, 85].

A.1 10d Super Yang Mills (SYM)

(9 + 1)-dimensional $U(N)$ $\mathcal{N} = 1$ super Yang-Mills action with metric $\eta_{\mu\nu} = \text{diag}(+1, -1, \cdots, -1)$ ($\mu, \nu$ runs from 0, 1, $\cdots$, 10) is given by,

$$ S = \frac{1}{g_Y^2} \int d^{10}x \left[ -\frac{1}{4} F_{\mu\nu}^a F^{\mu\nu a} + \frac{i}{2} \bar{\Psi}^a \Gamma^\mu D_\mu \Psi^a \right] $$

(A.1)

where \(^1\),

$$ F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu - i[A_\mu, A_\nu] $$

$$ D_\mu \Psi = \partial_\mu \Psi - i[A_\mu, \Psi] $$

$$ A_\mu = \mathbb{1} A_\mu^0 + T^a A_\mu^a $$

$$ \Psi = \mathbb{1} \Psi^0 + T^a \Psi^a $$

(A.2)

The gauge potential $A_\mu$ and Majorana-Weyl spinor $\Psi$ are in the adjoint representation of the gauge group $U(N)$. $T^a$ is the generator of the $SU(N)$ part of the gauge

\(^1\)Summation over repeated index implied unless otherwise stated
Appendix A. Dimensional Reduction of 10-dimensional Super-Yang-Mills Theory

group satisfying \( [T^a, T^b] = if^{abc}T^c \), where \( f^{abc} \) is completely anti-symmetric structure constants. \( \mathbb{1} \) is the \( N \times N \) Identity matrix and \( (A_\mu^0, \Psi^0) \) represents \( U(1) \) part of \( U(N) \). In the following analysis we will suppress the \( U(N) \) indices. The \( 32 \times 32 \) Dirac matrices \( \Gamma_\mu (\mu = 0, \cdots , 9) \) satisfies \( \{ \Gamma_\mu, \Gamma_\nu \} = 2\eta_{\mu\nu} \). The supersymmetry transformations that leaves the action invariant is given by,

\[
\delta A_\mu = i\epsilon \Gamma_\mu \Psi \\
\delta \Psi = \frac{1}{2} F_{\mu\nu} \Gamma^{\mu\nu} \epsilon
\] (A.3)

where \( \epsilon \) is an constant anticommuting Majorana-Weyl spinor and \( \Gamma_{\mu\nu} = \frac{1}{2}[\Gamma_\mu, \Gamma_\nu] \).

We will rewrite the gamma matrices in the following form,

\[
\Gamma_0 = \Gamma^0 = \mathbb{1} \otimes \sigma_2 \\
\Gamma_i = -\Gamma^i = \gamma_i \otimes i\sigma_1 \\
\Gamma_5 = \Gamma^0 \cdots \Gamma^9 = \mathbb{1} \otimes \sigma_3
\] (A.4)

where, \( \gamma_i \) are 9 real symmetric \( 16 \times 16 \) matrices satisfying \( \{ \gamma_i, \gamma_j \} = 2\delta_{ij} \). \( \mathbb{1} \) is the \( 16 \)-dimensional identity matrix and \( \sigma_{1,2,3} \) are the usual Pauli matrices,

\[
\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} ; \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} ; \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}
\] (A.5)

The first 8 \( \gamma \) matrices can be identified with the Dirac matrices of \( \text{spin}(8) \) and the last with \( 8 \) dimensional chirality. We will also use 16 component spinors \( \psi \) defined by

\[
\Psi = \psi \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}
\] (A.6)

The action now becomes,

\[
S = \frac{1}{g_{YM}^2} \int d^{10}x \, Tr \left[ -\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + \frac{i}{2} \psi^T D_0 \psi - \frac{i}{2} \psi^T \gamma_i D_i \psi \right]
\] (A.7)

For reduction to \( 0 + 1 \)-dimensions, all the fields are taken independent of all space directions. The volume term in the direction which the reduction is done is
Appendix A. Dimensional Reduction of 10-dimensional Super-Yang-Mills Theory

dropped. The action (eqn.(A.1)) becomes (\( T \) denotes transpose),

\[
S = \frac{1}{2g_{YM}^2} \int dt Tr \left[ (D_0 A_i)^2 + \frac{1}{2} [A_i, A_j]^2 + i\psi^T D_0 \psi - \psi \gamma_i [A_i, \psi] \right]
\]  

(A.8)

Let us now define \( A_i = \frac{X_i}{2\pi \alpha'} \) and \( \psi = \frac{\theta}{(2\pi \alpha')^2} \). The action becomes,

\[
S = \frac{1}{2g_s l_s} \int dt Tr \left[ (D_0 X_i)^2 + \frac{1}{2} \frac{1}{(2\pi \alpha')^2} [X_i, X_j]^2 + i\psi^T D_0 \psi - \frac{1}{(2\pi \alpha')^2} \psi \gamma_i [A_i, \psi] \right]
\]  

(A.9)

where, \( g_{YM}^2 = \frac{g_s^2}{16\pi^2 l_s^3} \). The above action is same as the one considered in BFSS matrix model (eqn.(2.1)).

A.2 From 10 to \( p + 1 \)

Consider all fields in 10d SYM action (eqn.(A.1)) independent of \( x_{p+1}, \cdots, x_{10} \). Let us consider the indices \( \mu, \nu, \cdots \) runs from 0, 1, \cdots, \( p \) and \( i, j, \cdots \) from \( (p + 1), \cdots, 10 \). Then the dimensional reduction gives,

\[
S = \frac{1}{g_{YM}^2} \int d^{(p+1)}x Tr \left[ \frac{1}{2} (D_\mu A_i)(D^\mu A_i) + \frac{1}{4} [A_i, A_j]^2 - \frac{1}{4} F_{\mu\nu} F^{\mu\nu} + \frac{i}{2} \psi^T D \psi - \frac{1}{2} \psi^T \gamma_i [A_i, \psi] \right]
\]  

(A.10)

where \( D = 1D_0 - \gamma_\mu D_\mu \) and 1 is \( 16 \times 16 \) identity matrix.
Consider dimensionally reduced maximally supersymmetric $U(N)$ Yang-Mills theory in 10-dimension to $(p+1)$ dimensions (Appendix A),

$$ S_{YM} = \int d^{p+1}L_{YM} = \frac{1}{2g_{YM}^2} \int d^{p+1}x \text{Tr}\{(D_{\mu}A_i)(D^{\mu}A_i) + \frac{1}{2}[A_i, A_j]^2$$

$$- \frac{1}{2}F_{\mu\nu}F^{\mu\nu} + i\theta^T D\theta - \theta^T \gamma_i [A_i, \theta]\} \quad (B.1)$$

where $D = 1D_0 - \sum_{j=1}^{p} \gamma_j D_j$ and $1$ is $16 \times 16$ identity matrix. $\eta_{\mu,\nu} = (1, -1, -1, \cdots, -1)$ and $\gamma_i$ ($i = 1, \cdots, 8$) are $16 \times 16$ Dirac matrices of spin(8). $\gamma_9 = \gamma_1 \cdots \gamma_8$ is the corresponding chirality matrix. The field content is given by the gauge potential $A_\mu$, 16 component Majorana-Weyl spinor $\theta$ and $10 - p$ scalars $A_i$, all in the adjoint representation of $U(N)$. Also, $D_{\mu}\bullet = \partial_{\mu} \bullet - i[A_\mu, \bullet]$ , where $\bullet$ can be replaced by any of $(A, \theta)$. We will review the background gauge fixing method in this case following [144].

Let us consider,

$$A_{\mu} = a_\mu + A'_{\mu}$$
$$A_i = a_i + A'_i$$
$$\theta = \Theta + \theta'$$ \quad (B.2)

Where $a_\mu, a_i, \Theta$ are background fields obeying classical equation of motion. Let us define a new covariant derivative as $\tilde{D}_\mu = \partial_\mu - i[a_\mu, \bullet]$. The primed fields are quantum fluctuations which is integrated out in the path integral to calculate
Appendix B. Background Gauge Fixing

The partition function. Also,

\[ F_{\mu\nu} = \bar{F}_{\mu\nu} + (\bar{D}_\mu A'_\nu - \bar{D}_\nu A'_\mu) - i[A'_\mu, A'_\nu] \tag{B.3} \]

where, \( \bar{F}_{\mu\nu} = \partial_\mu a_\nu - \partial_\nu a_\mu - i[a_\mu, a_\nu] \).

Now the allowed gauge transformation is the ones which keep the background unchanged, i.e. \( \delta a_\mu = \delta a_i = \delta \Theta = 0 \). Then the gauge transformation on the fluctuations are given by,

\[
\begin{align*}
\delta A'_\mu &= \bar{D}_\mu \alpha - i[A'_\mu, \alpha] \\
\delta A'_i &= -i[A'_i, \alpha] \\
\delta \theta' &= -i[\theta', \alpha] \tag{B.4}
\end{align*}
\]

The gauge fixing condition we use is,

\[ \bar{D}_\mu A'^\mu = 0 \tag{B.5} \]

Therefore the gauge fixing Lagrangian,

\[
\mathcal{L}_{gf} = -\frac{1}{2g_Y^2} Tr(\bar{D}_\mu A'^\mu)^2 \tag{B.6}
\]

where \( \xi \) is an arbitrary parameter (\( \xi = 0 \) gives Landau gauge and \( \xi = 1 \) Feynman gauge) and the ghost Lagrangian,

\[
\mathcal{L}_{gh} = \frac{1}{2g_Y^2} Tr\{(\bar{D}_\mu \tilde{\omega})(\bar{D}_\mu \omega - i[A'^\mu, \omega])\} \tag{B.7}
\]

Where \( \omega \) and \( \tilde{\omega} \) are ghost and anti ghost respectively. Consider the background such that \( a_i = 0 \), \( \Theta = 0 \) and \( \Omega = \bar{\Omega} = 0 \), where \( \Omega \) is the background for ghost. Let us consider the Lagrangian up to quadratic in quantum fluctuation or the 1-loop Lagrangian. We also use the classical equation of motion for the background fields.
Appendix B. Background Gauge Fixing

We get,
\[
\mathcal{L}_{YM}^{(1)} = \frac{1}{2g_Y^2} Tr\{(\bar{D}_\mu A'_\mu)^2 - (\bar{D}_\mu A_\mu')^2 - \frac{1}{2} \bar{F}_{\mu\nu} F^{\mu\nu} - iA'_\mu A_\nu' \bar{F}_{\mu\nu} + (\bar{D}_\mu A^{\mu'})^2 + i\theta'^T \bar{D} \theta'\}
\]
\[
\mathcal{L}_{gf}^{(1)} = -\frac{1}{2g_Y^2} Tr(\bar{D} A^{\mu'})^2
\]
\[
\mathcal{L}_{gh}^{(1)} = \frac{1}{2g_Y^2} Tr\{((\bar{D}_\mu \bar{\omega})'(\bar{D}^\mu \omega'))\}
\]

Then the full gauge fixed 1-loop Lagrangian in Feynman gauge (\(\xi = 1\)) is given by,
\[
\mathcal{L}^{(1)} = \frac{1}{2g_Y^2} Tr\{((\bar{D}_\mu A'_\mu)^2 - (\bar{D}_\mu A_\mu')^2 - \frac{1}{2} \bar{F}_{\mu\nu} F^{\mu\nu} - iA'_\mu A_\nu' \bar{F}_{\mu\nu} + i\theta'^T \bar{D} \theta' + (\bar{D}_\mu \bar{\omega})(\bar{D}^\mu \omega'))\}
\]
\[
= \frac{1}{2g_Y^2} Tr\{-A'_0(\bar{D})^2 A'_0 + A'_0(\bar{D})^2 A_0' - A_0(\bar{D})^2 A_0' + \bar{F}_{09}^2 + 2iA'_0 A_0' \bar{F}_{09} + i\theta'^T \bar{D} \theta' - \bar{\omega}'(\bar{D})^2 \omega')\}
\]

Now consider the case \(p = 1\) i.e. \((1+1)\)-dimensional SYM, then the above action becomes,
\[
\mathcal{L}^{(1)} = \frac{1}{2g_Y^2} Tr\{-A'_0(\bar{D})^2 A'_0 + A'_0(\bar{D})^2 A_0' - A_0(\bar{D})^2 A_0' + \bar{F}_{09}^2 + 2iA'_0 A_0' \bar{F}_{09} + i\theta'^T \bar{D} \theta' - \bar{\omega}'(\bar{D})^2 \omega')\}
\]

where \(\mu = 0, 9\) are the two dimensions of the SYM, \(\bar{D}^2 = \bar{D}_0^2 - \bar{D}_9^2\), and \(i = 1, \cdots, 8\). Let \(\theta' = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}\), where \(\theta_{1,2}\) are eight component spinors. Also consider
\[
1 = \begin{pmatrix} 1_{8\times8} & 0 \\ 0 & 1_{8\times8} \end{pmatrix}\quad \text{and} \quad \gamma_9 = \begin{pmatrix} 1_{8\times8} & 0 \\ 0 & -1_{8\times8} \end{pmatrix}.
\]
Then, \(\theta'^T \bar{D} \theta' = \theta_1^T (D_0 - D_9) \theta_1 + \theta_2^T (D_0 + D_9) \theta_2\) where, \(\theta_{1,2}^* = \theta_{1,2}\).

The Euclidean action is defined by \(S_E = -iS(t = -i\tau, A_0' = iA'_\tau)\) (where, path integral is defined as \(Z = \int \mathcal{D}[\bullet] e^{iS[\bullet]} = \int \mathcal{D}[\bullet] e^{-S_E[\bullet]}\) and \(\tau\) is considered real),
\[
S_E^{(1)} = -\frac{1}{2g_Y^2} \int d\tau Tr\{A'_\tau(\bar{D}_\tau)^2 A'_\tau + A'_\tau(\bar{D}_\tau)^2 A_{\tau}' + A_\tau(\bar{D}_\tau)^2 A_\tau' - \bar{F}_{\tau 9}^2 + 2iA'_{\tau} A_{\tau}' \bar{F}_{\tau 9} - \theta_1^T (\bar{D} - i\bar{D}_9) \theta_1 - \theta_2^T (\bar{D} + i\bar{D}_9) \theta_2 + \bar{\omega}'(\bar{D}_\tau)^2 \omega')\}
\]
where \((\bar{D}_E)^2 = (\bar{D}_r)^2 + (\bar{D}_g)^2\). Consider the case \(a_r = 0, a_g = constant\), then \(\bar{F}_{rg} = 0\) and the action becomes,

\[
S_E^{(1)} = -\frac{1}{2g_{YM}^2} \int d\tau \left\{ A'_i(\bar{D}_E)^2 A'_i + A'_r(\bar{D}_E)^2 A'_r + A'_g(\bar{D}_E)^2 A'_g - \theta'^T_1(\bar{D}_r - i\bar{D}_g)\theta'_1 - \theta'^T_2(\bar{D}_r + i\bar{D}_g)\theta'_2 + \bar{\omega}'(\bar{D}_E)^2 \omega' \right\}
\]

(B.12)

Which gives the 1-loop Euclidean partition function as (Note: as \(\theta\) and \(\theta^T\) are not independent, the path integral of the fermionic fluctuation will also give a factor of \(\frac{1}{2}\) like bosonic, but with a opposite sign),

\[
lnZ^{(1)} = \frac{10}{2} Tr(ln \bar{D}_E^2)_{bosonic} - \frac{8}{2} Tr(ln \bar{D}_L)_{fermionic} - \frac{8}{2} Tr(ln \bar{D}_R)_{fermionic} - Tr(ln \bar{D}_E^2)_{ghost} = 5Tr(ln \bar{D}_E^2)_{bosonic} - 4Tr(ln \bar{D}_E^2)_{fermionic} - Tr(ln \bar{D}_E^2)_{ghost}
\]

(B.13)

where \(\bar{D}_L\) and \(\bar{D}_R\) are operators acting on the left moving and right moving fermion, and \(\bar{D}_E^2 = \bar{D}_L \bar{D}_R\).
In this appendix Mathematica 7 codes used for various calculations for the second part of my thesis are given.

C.1 Fermion

Here $\omega$ is the frequency, $k$ is the momentum and $\mu = Qq$. $\epsilon$ is a small cut-off near $z = 0$. For zero temperature $Q = 2$.

C.1.1 Zero Temperature

The following is the Mathematica code used in computation of Fermion Green’s function calculation at zero temperature ($Q = 2$), (Greens0T[$\omega_-$, $k_-$, $\mu_-$, $\epsilon_-$] is the Green’s function at zero temperature, Greens0Tpre[$\omega_-$, $k_-$, $\mu_-$, $\epsilon_-$] is the same code but with more precision. Here the precision was set to 20 for the input variables, which can be altered by changing the parameter in SetPrecision[] function
and simultaneously changing the parameter WorkingPrecision in NDSolve.)

\[
\text{Greens0T}[\omega_, k_, \mu_, \epsilon_]:=
\text{Module}[\{kt = \omega, kx = k, mu = \mu, emod = \epsilon, f, z, Sol, G, bc0, bc1, \epsilon1 = 10^{-3}\epsilon\},
(*\epsilon1 is a small number in bc0*)
\text{f}[z_] := -1 - z^2 + 2z^2 \text{Log}[z];
bc0 = I\text{Sqrt}\left[\frac{mu^2}{2} - kx^2 + \epsilon1/(mu/Sqrt[2] + kx)\right];
(*exactly at } \omega = 0 \text{ the b.c. is given by bc0 otherwise it is bc1 = I ,I } = \sqrt{-1}^*\)
bc1 = I(1 - KroneckerDelta[kt, 0]) + KroneckerDelta[kt, 0] bc0; Sol =
\text{NDSolve}\{\text{f}[z]G'[z] + G[z]^2(kt + mu \text{Log}[z] - kx \text{Sqrt}[\text{f}[z]]) + (kt + mu \text{Log}[z] + kx \text{Sqrt}[\text{f}[z]]) == 0, G[1 - \epsilon] == bc1\}, G, \{z, \epsilon, 1 - \epsilon\},
\text{MaxSteps } \rightarrow \text{Infinity}; \text{Evaluate}[G[\epsilon]]/._\text{Sol}[[1]]\]
\( \omega \) and \( k \) given by \( \Delta \omega = \Delta k = \epsilon \) \( \sim 10^{-4} \). We have set \( \epsilon = 10^{-4} \) and \( \mu = 1 \).

\[
\text{epsilon} = 10^{-4}; \text{fok}[x\_, y\_] := \text{Im}[\text{Greens0T}[x, y, 1, 10^{-4}]]; \\
\text{fop}[x\_, y\_] := (\text{fok}[x + \epsilon, y] - \text{fok}[x, y]) / \epsilon; \text{flk}[x\_, y\_] := \\
(\text{fok}[x, y + \epsilon] - \text{fok}[x, y]) / \epsilon; \text{fokpr}[x\_, y\_] := \text{Im}[\text{Greens0Tpr}[x, y, 1, 10^{-4}]]; \\
\text{foppr}[x\_, y\_] := (\text{fokpr}[x + \epsilon, y] - \text{fokpr}[x, y]) / \epsilon;
\]

\( \text{maxo}[] \) finds the maximum of \( \text{fok}[\omega, k] \) with respect to \( \omega \) at a fixed \( k \). \( \text{maxk}[] \) finds the maximum of \( \text{fok}[\omega, k] \) with respect to \( k \) at a fixed \( \omega \). The output of these functions are as follows: \{\( \omega, k, \text{fokmax}, \text{fokmaxp} \}\), where \( \omega \) or \( k \) gives the position of the maximum depending on which is kept fixed. \( \text{fokmax} = \text{fok}[\omega, k] \) is the value at the maximum and \( \text{fokmaxp} = \text{fop}[\omega, k] \) or \( \text{flk}[\omega, k] \) is the value of the respective derivative at maximum. The loops for searching maximum starts at an initial guess for the position \( \text{xstart} \) or \( \text{ystart} \) and terminates when either \( \text{fokmax} > \text{fmax} \) or \( \text{fokmaxp} < \text{delta} \). \( \text{fmax} \) and \( \text{delta} \) are given as input. We have to also give \( \text{astart} \) as input, which determines the initial step size for maximum search in the loop.

\[
\text{maxo}[k\_, \text{astart}\_, \text{xstart}\_, \text{delta}\_, \text{fmax}\_] := \\
\text{Module}\{\{x, y, a\}, x = \text{xstart}; y = k; a = \text{astart}; \text{Label}[1]; \\
\text{While}[\text{fop}[x, y] > \text{delta} \&\& \text{fok}[x, y] < \text{fmax}, x += a]; a = .5a; \\
\text{If}[\text{Abs}[\text{fop}[x, y]] < \text{delta} \&\& \text{fok}[x, y] > \text{fmax}, \text{Goto}[\text{end}]]; \\
\text{While}[\text{fop}[x, y] < \text{delta} \&\& \text{fok}[x, y] < \text{fmax}, x += -a]; \\
a = .5a; \text{If}[\text{Abs}[\text{fop}[x, y]] < \text{delta} \&\& \text{fok}[x, y] > \text{fmax}, \text{Goto}[\text{end}], \text{Goto}[1]]; \\
\text{Label}[\text{end}]; \{x, y, \text{fok}[x, y], \text{fop}[x, y]\}
\]
maxk[omega__, astart__, ystart__, delta__, fmax__] :=
Module[{x, y, a}, x = omega;
  y = ystart; a = astart; Label[1];
  If[Abs[a] < 10^-5, Goto[end]]; While[fkp[x, y] > delta
    && fok[x, y] < fmax, y += a]; a = .5a;
  If[Abs[fkp[x, y]] < delta || fok[x, y] > fmax, Goto[end]]; 
  While[fkp[x, y] < delta && fok[x, y] < fmax, y -= a]; a = .5a;
  If[Abs[fkp[x, y]] < delta || fok[x, y] > fmax, 
    Goto[end], Goto[1]]; Label[end];
  {x, y, fok[x, y], fkp[x, y]}]

The module maxkloop[] searches for maximum of fok[ω, k] for a range of ω between ωst and ωe and at a interval int. yst corresponds to the initial guess for position of maximum along k at ω = ωst. ast is the value of astart used in previous program.

maxkloop[ωst__, astart__, ystart__, ωe__, int__] :=
Module[{x, y, a, data, Nsteps, i = 1}, x = ωst; y = yst;
  a = ast; Nsteps = Abs[Round[(ωe - ωst)/int]]; For[i = 1, i < Nsteps, i++,
    d[i] = maxk[x, a, y, 10^-4, 10^6]; x = x + int; y = N[d[i]][[2]]]; Table[d[i], {i, 1, Nsteps - 1}]]

C.1.2 Finite Temperature

The following is the Mathematica code used in computation of Fermion Green’s function calculation at finite temperature (Q ≠ 2). The module PrecisionSet[x, pre] sets precision of x to desired value pre. The module GreensFiniteTpre[ω, k, Q, q, ϵ, pre] calculates the Green’s function at finite temperature. ϵ sets the boundary cut-off near z = 0. fokpr[] is the imaginary part of the Green’s function at q = \frac{1}{2}, Q = 1.99 and pre = 25. Analysis at various values of temperature are done by changing Q at fixed q. fkppr[] evaluates derivative of fokpr[] w.r.t. k at fixed ω using “finite difference approximation” (ε) as in zero temperature case.
Appendix C. Numerical Programming

\begin{verbatim}
PrecisionSet[x_, pre_] := If[Precision[x] == Infinity, N[x, pre], SetPrecision[x, pre]];
GreensFiniteTpre[ω_, k_, Q_, q_, ϵ_, pre_] :=
Module[{kt = PrecisionSet[ω, pre], kx = PrecisionSet[k, pre], c1 =
PrecisionSet[Q, pre], c2 = PrecisionSet[q, pre], ϵmod = ϵ, f, z, Sol, G, sg, T},
  f[z_] := 1 - z^2 + ((z^2 c1^2)/2)Log[z]; Sol =
  NDSolve[{f[z] G'[z] + G[z]^2 (kt + c1 c2 Log[z] - kx Sqrt[f[z]]) +
    (kt + c1 c2 Log[z] + kx Sqrt[f[z]]) == 0,
    G[1 - ϵ] == 1}, G, {z, ϵ, 1 - ϵ},
  MaxSteps -> Infinity, WorkingPrecision -> N[pre - 1]]; sg = Evaluate[G[ϵ]] / Sol[[1]]];
  fokpr[x, y, ϵNd_] := Im[GreensFiniteTpre[x, y, 199/100, 1/2, ϵNd, 25]];
  fkppr[x, y, ϵ, ϵNd_] := (fokpr[x, y + ϵ, ϵNd] - fokpr[x, y, ϵNd]) / ϵ

The module maxkpr[] is similar to what described for zero temperature (maxk[]),
with a slight modification, which alters the value of ϵ (parameter of finite difference
approximation) according to the step size in the loop. maxklooppr[] is similar to
\end{verbatim}
the module used in zero temperature case (maxkloop[]).

\[
\text{maxkpr[omega, astart, ystart, delta, fmax, epsilonNd]} :=
\]
\[
\text{Module[\{x, y, a, \epsilon\}, x = omega; y = ystart; a = astart; Label[1]; If[Abs[a] < delta, Goto[end]];}
\]
\[
\text{If[Abs[a] > 10^{-6}, \epsilon = 10^{-10}, \epsilon = a10^{-4}];}
\]
\[
\text{While[FKPR[x, y, \epsilon, epsilonNd] > delta \&\& FKPR[x, y, \epsilon, epsilonNd] < fmax, y+=a]; a = a/2;}
\]
\[
\text{If[Abs[a] > 10^{-6}, \epsilon = 10^{-10}, \epsilon = a10^{-4}]; If[Abs[FKPR[x, y, \epsilon, epsilonNd]] < delta[FKPR[x, y, \epsilon, epsilonNd] > fmax, Goto[end], Goto[1]];}
\]
\[
\text{If[Abs[a] > 10^{-6}, \epsilon = 10^{-10}, \epsilon = a10^{-4}]; Label[end]; \{omega,}
\]
\[
\text{N[y, Abs[Log[10, delta]] + 1], FKPR[x, y, epsilonNd],}
\]
\[
\text{FKPR[x, y, \epsilon, epsilonNd]}\}
\]

\[
\text{maxkloop[omega, astart, ystart, omega, int]} :=
\]
\[
\text{Module[\{x, y, a, d, data, Nsteps, i = 1\}, x = omega; y = ystart; a = astart;}
\]
\[
\text{Nsteps = Abs[Round[(omega - ystart)/int]];}
\]
\[
\text{For[i = 1, i < Nsteps + 1, i++, d[i] = maxkpr[x, a, y, 10^8 - 7, 10^8 - 7, 10^8 - 10]; x = x + int;}
\]
\[
\text{y = d[i][2]]; Table[d[i], \{i, 1, Nsteps\}]}
\]

\text{Rootfindpr[omega, xi, f0, delta, epsilonNd], finds solution for the equation of the form FKPR[omega, k, \epsilon, epsilonNd] ==}
\[
\text{f0 at a given omega. k = xi is the initial guess for the solution. delta sets the allowed error in the solution. epsilonNd is the boundary cut-off in the Green's function.}
\]

\text{Rootfindpr[omega, xi, f0, delta, epsilonNd] :=
\]
\[
\text{Module[\{x = xi, \Delta\}, While[True, \Delta = N[(f0 - FKPR[omega, x, \epsilon, epsilonNd])]/FKPR[omega, x, 10^8 - 10, \epsilon, epsilonNd];}
\]
\[
\text{(*Precision is MachinePrecision*)}
\]
\[
\text{If[Abs[\Delta] > delta, x = x + \Delta, Break[]]]; x}
\]

\text{FWHMpr[omega, x1i, x2i, f0, delta, epsilonNd], finds the Full-Width-at-Half-Maximum (FWHM)}
in the $k$-direction ($\delta k$) for a given $\omega$. It uses the Rootfindpr[] function to determine two values of $k$ for which the value of the function fokpr[] has half of its maximum value ($f0 = \frac{f_{\text{max}}}{2}$, where $f_{\text{max}}$ is the maximum value). The initial guesses for the two values are $x1i$ and $x2i$. For a range of $\omega$, the data set for peak position and peak values are first obtained using maxkloopp[] and the output is stored as maxkdata. FWHMloop[] uses the maxkdata list to calculate the FWHM for the range of $\omega$.

\[
\text{FWHMpr}[\omega_, x1i_, x2i_, f0_, \delta_, \epsilon N d_]:=
\]
Module[{x1, x2, width},
        x1 = Rootfindpr[\omega, x1i, f0, \delta, \epsilon N d];
        x2 = Rootfindpr[\omega, x2i, f0, \delta, \epsilon N d];
        width = Abs[x1 - x2];
        {x1, x2, width}]

\[
\text{FWHMloop}[\omega st_, y1st_, y2st_, \omega e_, \text{int}_]:=
\]
Module[{x = \omega st, y1 = y1st, y2 = y2st, d, data, Nsteps, i = 1},
        Nsteps = Abs[Round[(\omega e - \omega st)/\text{int}]];
        For[i = 1, i < Nsteps + 1, i++,
            d[i] = FWHMpr[x, y1, y2, maxkdata[[i]][[3]]]/2, 10^−7, 10^−10];
            x = x + \text{int};
            y1 = d[i][[1]]; y2 = d[i][[2]];]
        Print[i, d[i]]; Table[maxkdata[[i]][[1]](*\omega *),
        {i, 1, Nsteps}]
]

C.2 Vector

Solution near boundary, $z = \epsilon = ep$ ($\epsilon$ is a small number, close to zero) for gauge field perturbation is given by $a(z) \simeq a^+ + a^- Log(z)$. So, $am = a^- = za'(z)\bigg|_{z=\epsilon}$ and $ap = a^+ = [a(z) - a^- Log(z)]\bigg|_{z=\epsilon}$. 

148
C.2.1 Finite Temperature

The module Cond[\(\omega, q, ep\)] gives Conductivity at finite temperature, where \(q\) is the square of the black-hole charge.

\[
\text{Cond}[\omega_, q_, ep_]:= \text{Module}[(\{Q = q, \epsilon = ep, eqn, kt = \omega, \nu, f, At, Je, Bc, Bc1, L, E1, H1, am, ap, F, r, S, x\},
\text{Clear}[d*]; f[r_]:= 1 - r^2 + (Q r^2/2) \text{Log}[r]; T = (1 - Q/4)/(2\pi); \nu = I \omega/(4\pi T);
\text{d}[0] = 1; n = 5; eqn = r f'[r] F''[r] + f[r]((2\nu + 1) r f'[r] + f[r]) F'[r] +
(r \nu(\nu - 1) f'[r]^2 + \nu r f''[r] f[r] +
(f[r] + r f'[r]) \nu f'[r] + r (kt^2 - Q f[r])) F[r]; Je[r_]:= \text{Sum}[d[j](1 - r)^j, \{j, 0, n\}];
(* Je[r] is the near horizon (r=1) solution ansatz for the equation \text{eqn}==0*)
L = \text{Table}[\text{FullSimplify}[\text{SeriesCoefficient}[\text{eqn}./F \rightarrow \text{Je}, \{r, 1, i\}], \{i, n\}];
(*L[i][i]==0 is the equations obtained by putting the ansatz (Je) in the expression (eqn),
then expanding in the power series about r = 1, and subsequently
setting the coefficients of each power to zero *)
\text{For}[i = 1, i < n + 1, i++, \text{Si} = d[i]/. \text{Solve}[L[[i]] == 0, d[i]][[1]]; d[i] = \text{Si};]
(* The unknown series coeff of the ansatz, d[i][i] are obtained solving the coupled equations L*)
\text{Bc}[r_]:= \text{Sum}[d[j](1 - r)^j, \{j, 0, n\}; \text{Bc1}[r_]:= D[\text{Bc}[x], x]/. x \rightarrow r;
(*Bc[x], and Bc1[x]=Bc'[x] are solutions of eqn near horizon, and used for boundary condition*)
\text{E1} = \text{NSolve}[(\text{eqn} == 0, F[1 - \epsilon] == \text{Bc}[1 - \epsilon]), F'[1 - \epsilon] == \text{Bc1}[1 - \epsilon]),
F, \{r, \epsilon, 1 - \epsilon\}, \text{MaxSteps} \rightarrow 10^7]; H1[r_]:= f[r]^\nu \text{Evaluate}[F[r]]/. \text{E1}[[1]][[1]]; am = r D[H1[r], r]/. r \rightarrow \epsilon; \text{ap} = H1[\epsilon] - am \text{Log}[\epsilon]; -I/(kt)(ap/am)]
C.2.2 Zero Temperature

The module Cond[\(\omega, ep\)] gives Conductivity at zero temperature,

\[
\text{CondT0}[kt\_, ep\_] := \text{Module}[[\{\epsilon = ep, eqn, eqn1, \omega = kt, T, f, At, Je, Bc, Bc1, E1, H1, am, ap, g, r, S, x, n, i, L, d\}],
\]

Clear[d*]; \(f[r\_] := 1 - r^2 + 2r^2 \log[r]; d[0] = 1; n = 5; eqn1 = g''[r] + \left(1/r + f'[r]/f[r]\right)I(\omega/(1 - r)^2 + I(\omega/(3(1 - r)))g'[r] + (i \omega/2(1 - r)^2)/r - \omega^2/4(1 - r)^4 + \omega^2/f[r]^2 - 8/f[r]) +
\]

\[\frac{i(\omega(\frac{2}{1-r} + f'[r]/f[r]))}{2(1 - r)^2} + I(\omega/6(I(\omega/6 + 1)(1/(1 - r)^2)) +
\]

\[I(\omega/(6(1 - r))(1/r + f'[r]/f[r] + I(\omega/(1 - r)^2))g'[r];
\]

\(eqn = (1 - r)^3 eqn1; Je[r\_] := \text{Sum}[d[j](1 - r)^j, \{j, 0, n\}];
\]

\(L = \text{Table}\left[\text{FullSimplify}[\text{SeriesCoefficient}[eqn/.g \to \text{Je}, \{r, 1, i\}], \{i, n\}\right];
\]

\(For[i = 1, i < n + 1, i++, S_i = d[i]/.\text{Solve}\left[L[[i]] == 0, \text{d}[i] \right][[1]]; \text{d}[i] = S_i;\)
\]

\(Bc[r\_] := \text{Sum}[d[j](1 - r)^j, \{j, 0, n\}];\)

\(Bc1[r\_] := D[Bc[x], x] / . x \to r;\)

\(E1 = \text{NSolve}\left[\{\epsilon = 0, g[1 - \epsilon] == \text{Bc}[1 - \epsilon], g'[1 - \epsilon] == \text{Bc1}[1 - \epsilon], g, \{r, \epsilon, 1 - \epsilon\}, \text{MaxSteps} \to \text{Infinity}\right]; H1[r\_] := \text{Exp}[I(\omega/(2(1 - r)))((1 - r)^(-I(\omega/6)))]\text{Evaluate}[g[r]] / . E1[[1]][[1]];\)

\(\text{am} = r \times D[H1[r], r] / . r \to \epsilon; \text{ap} = H1[\epsilon] - \text{am} \log[\epsilon];\)

\(- I/(\omega)(\text{ap}/\text{am})\]
C.3 Scalar

C.3.1 Finite Temperature

The following is the Mathematica code used in Scalar Green’s function calculation at finite temperature,

```
Solution[N_, epsilon_, omega_, momentum_, charge_] :=
Module[{nt = N, w = SetPrecision[omega, 30], ep = SetPrecision[epsilon, 30],
    k = SetPrecision[momentum, 30],
    Q = SetPrecision[charge, 30], P, delta, A, B, V, S, L, HorizonSeries, HorizonSeriesPrime},
Clear[b*, a*];
SetPrecision[a*, 30]; SetPrecision[b*, 30]; T = 1/(4Pi)(2 - Q^2/2);
SetPrecision[T, 30]; m = 1/2; SetPrecision[m, 30];
p = SetPrecision[1 + Sqrt[1 + m^2], 30]; delta[z_] := 1 - z^2 + z^2/2Q^22Log[z];
P[z_] := 1 - z^2 + z^2Q^22Log[z]/2; A[z_] := -z^2P[z]^2;
B[z_] := zP[z](z^2(Q^2/2 - 1) + z^2Q^22Log[z]/2 - 1); V[z_, w_] :=
- P[z]k^2z^2 + z^2(w + QLog[z])^2 + m^22P[z]; S[z_] :=
Sum[a[n](z - 1)^n, {n, 0, nt}];

(*Series expansion near horizon*)
d = -Iw/(4PiT); SetPrecision[d, 30]; L[z_, w_] :=
Series[D[S[z], {z, 2}]delta[z]^2A[z] + D[S[z], z](B[z]delta[z]^2 +
2ddelta[z]D[delta[z], z]A[z]) + S[z](delta[z]^2V[z, w] +
A[z]d(d - 1)D[delta[z], z]A[z] + dB[z]delta[z]D[delta[z], z] +
ddelta[z]D[delta[z], {z, 2}]A[z]), {z, 1, nt}];

(*Finding solutions for series coefficients a[i]*)
b[0] = a[0]; For[i = 3, i < nt + 1, i++, b[i - 2] = a[i - 2]/.Solve[SeriesCoefficient[L[z, w], {z, 1, i}]
== 0, a[i - 2]][[1]]; a[i - 2] = b[i - 2];
```

151
Appendix C. Numerical Programming

(*Writing Series at horizon*)

HorizonSeries[z_, w_] = Sum[b[n](z - 1)^n, {n, 0, nt - 2}];
HorizonSeriesPrime[z_, w_] = D[HorizonSeries[z, w], z];

(*Solving the differential equation numerically*)

a[0] = 1; zb = ep;
Final = NDSolve[{PhiEqn == 0, G[1 - ep] == HorizonSeries[1 - ep, w], G'[1 - ep] == HorizonSeriesPrime[1 - ep, w]}, G, {z, zb, 1 - ep}, WorkingPrecision -> 25, MaxSteps -> Infinity];
Source[z_] = delta[z]^4 G[z]z^{p - 2}; Fluctuation[z_] = D[Source[z], z]z^{(3 - 2p)/(2p - 2)}; gw = Fluctuation[zb]/Source[zb]/Final]

C.3.2 Zero Temperature

The following is the Mathematica code used in Scalar Green’s function calculation at zero temperature,

nt = 7; (* number of terms *)
ep = 10^(-5); SetPrecision[ep, 30]; Q = 2; SetPrecision[Q, 30]; m = 1/2;
SetPrecision[m, 30]; d = I - I w/6; p = SetPrecision[1 + Sqrt[1 + m^2], 30];
P[z_] := 1 - z^2 + z^2Q^2Log[z]/2; A[z_] := z^2P[z]^2; B[z_] := zP[z](z^2(Q^2/2 - 1) + z^2Q^2Log[z]/2 - 1); V[z_, w_] := -P[z]k^2z^2 + z^2(w + QLog[z])^2 - m^22P[z];
S[z_] := Sum[a[n](z - 1)^n, {n, 0, nt}];
Appendix C. Numerical Programming

\((\text{Series expansion near horizon})\)
\[
L[z_-, w_-] := \text{Series}[4S[z]V[z, w](-1 + z)^4 +
2B[z](-1 + z)^2((Iw + 2d(-1 + z))S[z] + 2S'[z](-1 + z)^2) +
A[z]((-w(w + 4I(-1 + z)) + 4d(1 + Iw - z)(-1 + z) +
4d^2(-1 + z)^2)S[z] +
4(Iw + 2d(-1 + z))(-1 + z)^2S'[z] + 4(-1 + z)^2S''[z]), \{z, 1, nt\}]; b[0] = a[0];
\]
\[
\text{For}[i = 6, i < nt + 1, i++, b[i - 5] = a[i - 5]/. \text{Solve}[\text{SeriesCoefficient}[L[z, w], \{z, 1, i\}] == 0, a[i - 5]][[1]]; a[i - 5] = b[i - 5]];}

\((\text{Writing Series at horizon})\)
\[
\text{HorizonSeries}[z_-, w_-] = \text{Sum}[b[n](z - 1)^n, \{n, 0, nt - 5\}];
\text{HorizonSeriesPrime}[z_-, w_-] = D[\text{HorizonSeries}[z, w], z];
\]

\((\text{Solving the differential equation numerically})*\)
\[
w = 0.1; \text{a[0]} = 1; \text{zb = ep; PhiEqn := 4G[z]V[z, w](-1 + z)^4 +}
2B[z](-1 + z)^2((Iw + 2d(-1 + z))G[z] + 2G'[z](-1 + z)^2) +
A[z]((-w(w + 4I(-1 + z)) + 4d(1 + Iw - z)(-1 + z) +
4d^2(-1 + z)^2)G[z] +
4(Iw + 2d(-1 + z))(-1 + z)^2 G'[z] + 4(-1 + z)^2G''[z]);
\]
\[
\text{Final[\text{omega}, \text{momentum}]} := \{w = \text{SetPrecision}[\text{omega}, 30];
\text{k = SetPrecision[momentum, 30]; \text{NDSolve}[\{\text{PhiEqn} == 0, G[1 - ep] ==}
\text{HorizonSeries}[1 - ep, w], G'[1 - ep] == \text{HorizonSeriesPrime}[1 - ep, w] \}, G, \{z, zb, 1 - ep\},
\text{WorkingPrecision -> 25, MaxSteps -> Infinity} \});
\text{Source[z_] = \text{Exp}[I w/(2(1 - z))](z - 1)^\text{dG}[z]z^p(p - 2);}
\text{Fluctuation[z_] = D[Source[z], z]z^3 - 2p)/(2p - 2);}
gw = \text{Fluctuation[zb]/Source[zb]};
\]

153