NUMERICAL SIMULATIONS STATISTICS INVOLVING EIGENVALUES OF RANDOM MATRIX MODELS

This document provides a brief overview of the Hermitian one-matrix model and the Hermitian two-matrix model, and gives some numerical simulations associated to both models.

1. The Hermitian one-matrix model

The Hermitian one-matrix model is the space of $N \times N$ Hermitian matrices $H(N)$ equipped with the probability distribution

$$d\tilde{P}^{(N)}(H_N) = \tilde{Z}_N^{-1} e^{-Tr(V(H_N))} dH_N.$$

Here, V is an even degree polynomial with positive leading coefficient, and dH_N is Lebesgue measure on the entries of H_N.

$$dH_N = \prod_{i=1}^{N} dh_{ii} \prod_{i<j} dh_{ij}^R \prod_{i<j} dh_{ij}^I.$$

Here, h_{ij} is entry (i,j) of H_N and h_{ij}^R, h_{ij}^I denote the real and imaginary parts of h_{ij} respectively. The quantity $\tilde{Z}_N$ is a normalization constant, making $\tilde{P}^{(N)}$ a probability measure. If $V(x) = tx^{2m}$, the eigenvalues of the one-matrix model are distributed on the interval $[-C_{t,m}N^{\frac{1}{2m}}, C_{t,m}N^{\frac{1}{2m}}]$ for some constant $C_{t,m}$ [Dei99]. Therefore, it is helpful to scale the matrices by an appropriate factor so that the support of the distribution is bounded for large N. The *scaled* probability distribution for the one-matrix model is

$$d\mathcal{P}^{(N)}(H_N) = Z_N^{-1} e^{-NTr(V(H_N))} dH_N. \tag{1}$$

The spectral theorem for Hermitian matrices ([HJ90], Theorem 2.5.6) states that $H_N = U\Lambda U^*$, where U is unitary and Λ is the diagonal matrix of eigenvalues of H_N. This can be used to write $\mathcal{P}^{(N)}$ just in terms of the eigenvalues of H_N. Order the eigenvalues λ_i of H_N so that

$$\lambda_1 \leq \cdots \leq \lambda_N.$$

After performing a change of variables, the probability distribution can be rewritten [AGZ10]

$$d\hat{\mathcal{P}}^{(N)}(\lambda) = \hat{Z}_N^{-1} e^{-N\sum_{i=1}^{N} V(\lambda_i)} \prod_{i<j} (\lambda_i - \lambda_j)^2 d\lambda_1 \cdots d\lambda_N.$$

Notice that the change of variables here has altered the normalization constant and therefore the probability measure has been relabeled as $\hat{\mathcal{P}}^{(N)}$ instead of $\mathcal{P}^{(N)}$. $\hat{\mathcal{P}}^{(N)}$ is restricted to the sector $\lambda_1 \leq \cdots \leq \lambda_N$ in $\mathbb{R}^N$. One approach for studying these statistics is to consider

the following minimization problem: first rewrite the probability distribution so that all terms are in the exponent:

$$d\mathcal{P}^{(N)}(\lambda) = (N!Z_N)^{-1}e^{-N^2L_N^V(\lambda)}d\lambda_1\cdots d\lambda_N$$

where

$$L_N^V(\lambda) = \frac{1}{N}\sum V(\lambda_i) + \frac{1}{N^2}\sum_{i\neq j}\log\frac{1}{|\lambda_i-\lambda_j|}. \tag{2}$$

The main contribution to integrals with respect to this distribution should come from vectors

$$\vec{w}^* = (w_1^*,\ldots,w_N^*)$$

which minimize L_N^V. Let

$$\mu_{\vec{w}^*} = \mu_N = \frac{1}{N}\sum_{k=1}^{N}\delta_{w_k^*} \tag{3}$$

be the normalized counting measure for the vector $\vec{w}^*$. This measure is related to the eigenvalue distribution for the one-matrix model. Let

$$\mathcal{R}_1(\lambda_1) := N\int_{\mathbb{R}^{n-1}}\mathcal{P}^{(N)}(\lambda_1,\ldots,\lambda_N)d\lambda_2\cdots d\lambda_N. \tag{4}$$

This is called the 1-point correlation function for $\mathcal{P}_N$. This function is related to the eigenvalue distribution by the following formula [Dei99]:

$$\int_B\frac{1}{N}\mathcal{R}_1(\lambda_1)d\lambda_1 = \mathbb{E}\left(\frac{\#\text{ of eigenvalues in }B}{N}\right). \tag{5}$$

Theorem 1. [Dei99] *As N approaches ∞, $\mu_{\vec{w}^*}$ and $\frac{1}{N}\mathcal{R}_1(\lambda_1)$ (Equations (3) and (4) respectively) both converge weakly to the same measure.*

Theorem 1 reveals that studying the eigenvalue distribution is the same as studying the minimizing vector $\vec{w}^*$ of the function L_N^V, defined in Equation (2). To illustrate the eigenvalue distribution of the Hermitian one-matrix model, the entries of the minimizing vector $\vec{w}^*$ for L_N^V when $N = 1000$ and

$$V(x) = 0.1x^4 + x^2$$

are shown in the histogram in Figure 1.

The minimization algorithm chosen for this simulation is an interior point method. In general, to minimize a function $f(\vec{x})$ subject to constraints $c_i(\vec{x}) \geq 0$, the interior point method alters the original function f with a barrier function:

$$f_\mu(\vec{x}) = f(\vec{x}) - \mu\sum_i\log(c_i(\vec{x})).$$

Here, μ is a small constant which approaches zero. Minimizing f_μ when μ is small mimics the constrained original problem.

In order to minimize L_N^V subject to the constraint that $\lambda_1 < \ldots < \lambda_N$, the MATLAB

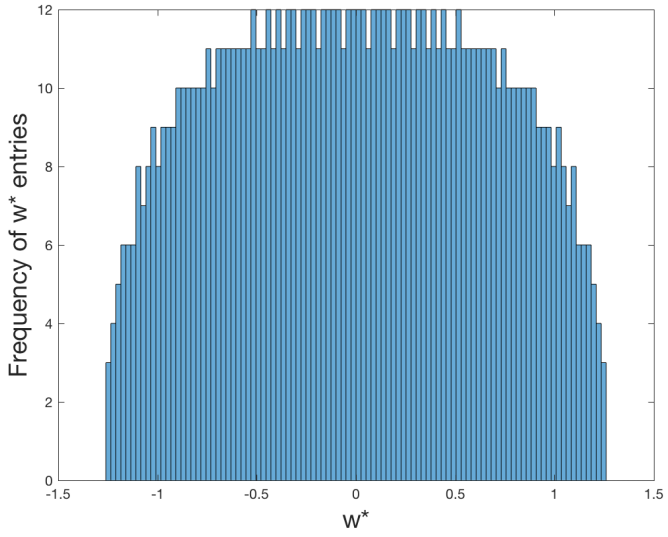

FIGURE 1. Histogram for the entries of the minimizing vector $\vec{w}^*$ for L_N^V when $N = 1000$ and $V(x) = 0.1x^4 + x^2$.

function "fmincon" is used, which by default uses an interior point method [MAT16]. The constraint given is $A\vec{\lambda} \leq \vec{0}$ where

$$A = \begin{pmatrix} 0 & 0 & \cdots & & \\ 1 & -1 & 0 & \cdots & \\ 0 & 1 & -1 & 0 & \ddots \\ 0 & 0 & \ddots & \ddots & \\ \vdots & & & & \\ & & & 1 & -1 \end{pmatrix}. \tag{6}$$

(This assures that for each $i = 2, \ldots, N-1$, $\lambda_{i-1} \leq \lambda_i$). After adjusting L_N^V with a barrier function, MATLAB then solves the approximate problem using Newton's method. The MATLAB code used is shown in Figure 2. The MATLAB code used to create Figure 1 is shown in Figure 3.

Remark 1. *It is possible that the minimizing vector λ has entries that are very close together so that $\lambda_i - \lambda_{i-1}$ is very small for some i. This would mean that the solution vector is very close to one of the boundary constraints, $\lambda_i = \lambda_{i-1}$. It is reasonable to question whether adding the barrier function affects the minimization significantly in this case, since it is designed to keep solutions away from the boundary. In order to address this issue, the minimizing vector found using fmincon was used as an initial vector for another minimization function, "fminunc," which uses the quasi-Newton method without*

```
N=1000;
temp=1:N;
x0=-1.2*ones(1,N)+(2.4/N)*temp;
x00=x0((N/2)+1:N);
R(x00)
A=sparse(1,1,1,N/2,N/2);
for i=2:(N/2)
    A(i,i-1)=1;
end
b=zeros(N/2,1);
options=optimoptions('fmincon','MaxFunctionEvaluations',100*N);
Aeq=[];
beq=[];
nonlcon=[];
minarg=fmincon(@R,x00,A-eye(N/2),b,Aeq,beq,zeros(N/2,1),2*ones(N/2,1),nonlcon,options);
xmin1000posOnematrix=minarg(1:(N/2));
new=R(xmin1000posOnematrix)
function poly1=V(x)
poly1=0.1*x^4+x^2;
end
function onematrix=R(xpos)
totallengthpos=length(xpos);
x=horzcat(-1*fliplr(xpos),xpos);
t=0.0001;
N=length(x);
firstterm=0;
for i=1:N
    firstterm=firstterm+V(x(i));
end
thirdterm=0;
for i=1:N
    for j=1:N
        if i~=j
            thirdterm=thirdterm+log(abs(x(i)-x(j)));
        end
    end
end
onematrix=(1/N)*firstterm-(1/N^2)*thirdterm;
end
```

FIGURE 2. MATLAB code used to minimize L_N^V with $N = 1000$ and $V(x) = 0.1x^4 + x^2$

any constraints. The result of the latter minimization changed the solution only minimally. Specifically, the minimum value of the function was the same up to the sixth decimal place, and the entries of the minimizing vector differed by at most 0.0007. This indicates that the barrier function does not significantly change the result found by fmincon.

```
load xmin1000posOnematrix
xtotal=horzcat(-1*fliplr(xmin1000posOnematrix),xmin1000posOnematrix);
xlength=length(xtotal);
figure(3)
histogram(xtotal,100)
xlabel('w*','fontsize',18)
ylabel('Frequency of w* entries','fontsize',18)
saveas(gcf,'1matrix100x4matlab.png')
```

FIGURE 3. MATLAB code used to create Figure 1

2. The Hermitian two-matrix model

The two-matrix model is the space of pairs of $N \times N$ Hermitian matrices (M_1, M_2) equipped with the probability distribution

$$\mathcal{P}^N = Z_N^{-1} e^{-Tr(V(M_1)+W(M_2)-2\tau M_1 M_2)} dM_1 dM_2. \tag{7}$$

Here, V and W are even degree polynomials, dM_i denotes Lebesgue measure on the entries of M_i, and $\tau \in \mathbb{R} \setminus \{0\}$ is a coupling constant. The large N behavior of the similarly defined one-matrix model has been characterized in full detail. But the techniques used to analyze the one-matrix model do not carry over to the two-matrix model completely: the interaction term $-2\tau M_1 M_2$ makes this model much more difficult to work with.

After diagonalizing the matrices M_1, M_2 and performing a change of variables, the interaction term is reduced to the following integral over the unitary group:

$$\mathcal{I}^{(N)}(X_N, Y_N, \tau) = \int_{U(N)} e^{2\tau Tr X_N U Y_N U^*} dU.$$

This integral is known as the Harish-Chandra, Itzykson, Zuber (HCIZ) integral. Results of Goulden, Guay-Paquet, and Novak in 2012 [GGPN14] and Guionnet and Novak in 2015 [GN15] on the HCIZ integral reveal that for small τ,

$$|\tau| < \frac{7}{38016}, \tag{8}$$

the probability distribution (7) can be rewritten (as in the one-matrix case) as a single exponent:

$$d\mathcal{P}^{(N)} = \hat{Z}_N^{-1} e^{-N^2 \hat{R}_{N,\tau}(x,y)} d^N X_N d^N Y_N$$

where

$$\begin{aligned} \hat{R}_{N,\tau}(x,y) := \left[\frac{1}{N}\sum_i V(x_i) + W(y_i)\right] + \frac{1}{N^2}\sum_{i,j} 2\tau x_i y_j (1 + \tau x_i y_j) + S(\tau) + \\ \frac{1}{N^2}\sum_{i \neq j} \log\left(\frac{1}{|x_i - x_j|}\right) + \log\left(\frac{1}{|y_i - y_j|}\right) \end{aligned} \tag{9}$$

and $S(\tau)$ is $O(\tau^3)$. Therefore, up to order τ^2, the leading contribution to integrals with respect to (7) is found by minimizing

$$R_{N,\tau}(\vec{x},\vec{y}) := \frac{1}{N}\sum_{i=1}^{N}(V(x_i)+W(y_i)) + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} 2\tau x_i y_j(1+\tau x_i y_j) + \frac{1}{N^2}\left(\sum_{\substack{i=1\\ i\neq j}}^{N}\sum_{j=1}^{N}\log|x_i-x_j|^{-1}+\log|y_i-y_j|^{-1}\right) \tag{10}$$

The function $R_{N,\tau}$ is convex in the region $\{(\tau, x_1^*,\ldots,x_N^*,y_1^*,\ldots,y_N^*) \ : \ |\tau| < \frac{\left(\sqrt{1+24\beta^2}-1\right)}{12\beta^2}\}$. Here, β is defined so that both collections of eigenvalues for the Hermitian one-matrix models associated to V and W are supported in $[-\beta,\beta]$. In this specific case, β can be found by adjusting the calculations slightly in [Erc11], section 2.4: when $V(x)=\epsilon x^4+x^2$,

$$\beta = 2\sqrt{z_0(\epsilon)} \tag{11}$$

where

$$z_0(\epsilon) = \frac{1-\sqrt{1+192\epsilon}}{-96\epsilon}. \tag{12}$$

The potentials chosen for this example are

$$V(x) = 0.1x^4 + x^2$$
$$W(y) = 0.2x^4 + y^2.$$

Therefore, setting $\epsilon = .1$ and $\epsilon = .2$ in Equations (11) and (12) yields $\beta \approx 1.206655$ and $\beta \approx 1.048505$ respectively. Therefore, choose $\beta = 1.21$ so that both μ^V and μ^W are supported on $[-\beta,\beta]$. Using this β, the restriction on τ in is

$$|\tau| < 0.285. \tag{13}$$

For τ satisfying this condition, a local minimization technique can be used to determine the minimizing vector in this region. The fmincon function in MATLAB was used again with constraints $A\vec{x} \leq \vec{0}$ and $A\vec{y} \leq \vec{0}$, where A is defined in Equation (6).

Remark 2. *As pointed out in Remark 1, it is possible that the solution vector $(\vec{x},\vec{y})$ has entries very close together, so that the minimizing vector is close to the boundary avoided by the barrier function. Once again, the solution vector found by fmincon was used as the input vector for the fminunc function. Just as with the one-matrix case, the solution did not change significantly: the minimum value of $R_{N,\tau}$ was the same up to the sixth decimal place, and the entries of the minimizing vector differed by at most 0.0006. This indicates that the barrier function does not significantly change the result found by fmincon.*

Figure 4 shows a histogram of pairs (x_i,y_j) of entries from minimizing vectors $\vec{x}$ and $\vec{y}$ of $R_{N,\tau}$ in the appropriate region. Here, τ is chosen to be 0.0001 (which satisfies both Equation (13) and (8)), and $N = 1000$.

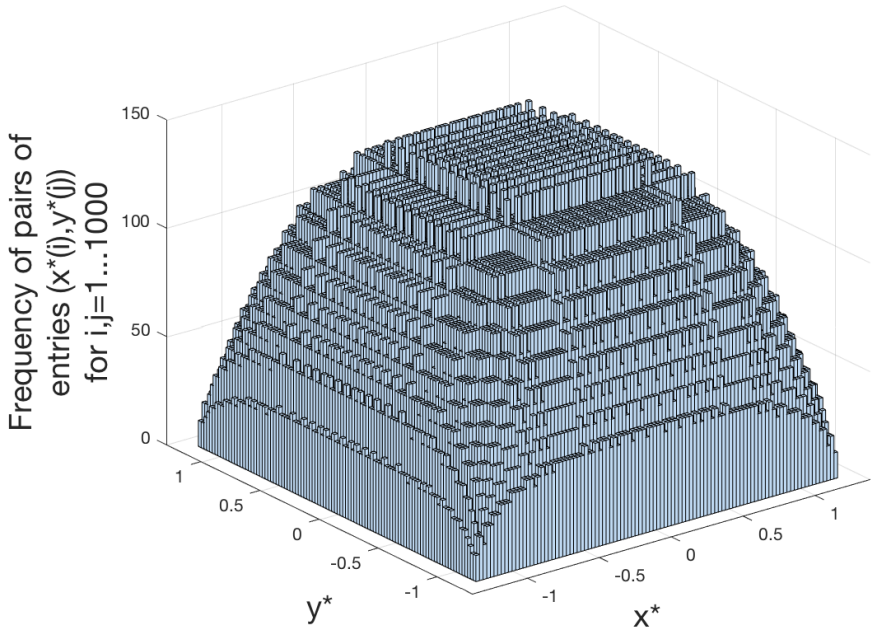

FIGURE 4. Histogram for the minimizing vector $(\vec{x}^*, \vec{y}^*)$ of $R_{N,\tau}(x, y)$ (Equation (10)) as pairs (x_i^*, y_j^*) with $N = 1000$, $V(x) = 0.1x^4 + x^2$, $W(y) = 0.2y^4 + y^2$, and $\tau = 0.0001$.

The code used to minimize $R_{N,\tau}$ is shown in Figure 5. The code used to create the histogram in Figure 4 is shown in Figure 6. The minimizing vector $(\vec{x}^*, \vec{y}^*)$ of $R_{N,\tau}(\vec{x}, \vec{y})$ (Equation (10)) is compared to the minimizing vectors for the corresponding one-matrix models with potentials V and W. Let $\vec{w}^*$ minimize L_N^V (Equation (2)), and $\vec{z}^*$ minimize L_N^W. A result of my dissertation states that for τ small, the ordered vector $\vec{x}^*$ should be approximated by the ordered vector $\vec{w}^*$ for large N. Figure 7 shows $\vec{x}^*$ plotted against $\vec{w}^*$. Similarly, for τ small, the ordered vector $\vec{y}^*$ should be approximated by the ordered vector $\vec{z}^*$ for large N. Figure 8 shows $\vec{y}^*$ plotted against $\vec{z}^*$. In both of the figures below, $N = 1000$, $\tau = 0.0001$, $V(x) = 0.1x^4 + x^2$, and $W(y) = 0.2y^4 + y^2$. The plotted points lie very close to the line $f(x) = x$ in both figures, indicating that $\vec{x}^*$ and $\vec{y}^*$ are very close to $\vec{w}^*$ and $\vec{z}^*$ respectively.

The MATLAB code used to create Figures 7 and 8 is shown in Figure 9

```
N=1000;
temp=1:N;
x0=-1.2*ones(1,N)+(2.4/N)*temp;
y0=-1.2*ones(1,N)+(2.4/N)*temp;
p00=horzcat(x0((N/2)+1:N),y0((N/2)+1:N));
R(p00)
A=sparse([1,(N/2)+1],[1,(N/2)+1],[1,1],N,N);
for i=2:(N/2)
    A(i,i-1)=1;
    A((N/2)+i,(N/2)+i-1)=1;
end
b=zeros(N,1);
options=optimoptions('fmincon','MaxFunctionEvaluations',100*N);
Aeq=[];
beq=[];
nonlcon=[];
minarg=fmincon(@R,p00,A-eye(N),b,Aeq,beq,zeros(N,1),2*ones(N,1),nonlcon,options);
xmin1000posConB=minarg(1:(N/2));
ymin1000posConB=minarg((N/2)+1:N);
new=R(horzcat(xmin1000posConB,ymin1000posConB))
function poly1=V(x)
poly1=0.1*x^4+x^2;
end
function poly2=W(y)
poly2=0.2*y^4+y^2;
end
function twomatrix=R(ppos)
totallengthpos=length(ppos);
xpos=ppos(1:(1/2)*totallengthpos);
ypos=ppos((1/2)*totallengthpos+1:totallengthpos);
p=horzcat(-1*fliplr(xpos),xpos,-1*fliplr(ypos),ypos);
totallength=length(p);
x=p(1:(1/2)*totallength);
y=p((1/2)*totallength+1:totallength);
t=0.0001;
N=length(x);
firstterm=0;
for i=1:N
    firstterm=firstterm+V(x(i))+W(y(i));
end
secondterm=0;
for i=1:N
    for j=1:N
        secondterm=secondterm+2*t*x(i)*y(j)*(1+t*x(i)*y(j));
    end
end
thirdterm=0;
for i=1:N
    for j=1:N
        if i~=j
            thirdterm=thirdterm+log(abs(x(i)-x(j)));
        end
    end
end
fourthterm=0;
for i=1:N
    for j=1:N
        if i~=j
            fourthterm=fourthterm+log(abs(y(i)-y(j)));
        end
    end
end
twomatrix=(1/N)*firstterm-(1/N^2)*secondterm-(1/N^2)*thirdterm-(1/N^2)*fourthterm;
end
```

FIGURE 5. MATLAB code used to minimize $R_{N,\tau}$

```
load xmin1000posConB
load ymin1000posConB
xtotal2=horzcat(-1*fliplr(xmin1000posConB),xmin1000posConB);
ytotal2=horzcat(-1*fliplr(ymin1000posConB),ymin1000posConB);
zs=zeros(xlength*ylength,2);
for i=1:xlength
    for j=1:ylength
        zs(i+(j-1)*ylength,1)=zs(i+(j-1)*ylength,1)+xtotal(i);
        zs(i+(j-1)*ylength,2)=zs(i+(j-1)*ylength,2)+ytotal(j);
    end
end
hist3(zs,[100,100])
```

FIGURE 6. MATLAB code used to create the histogram in Figure 4.

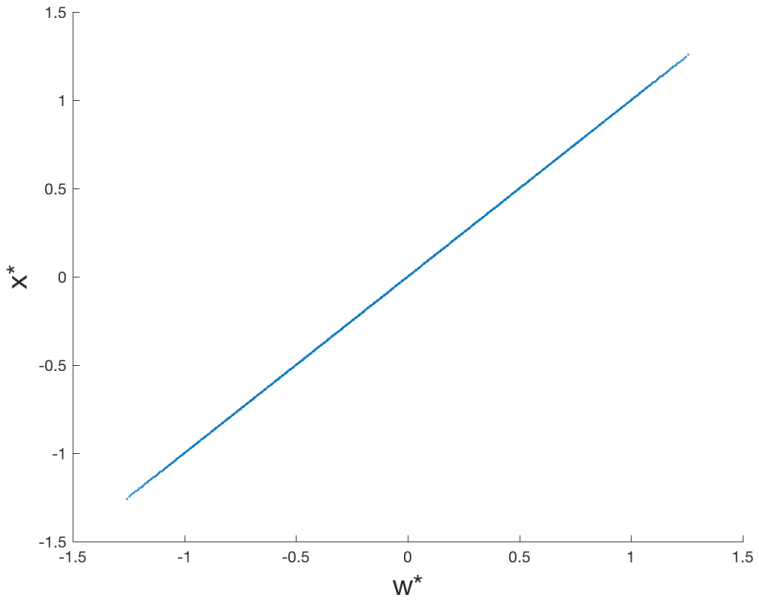

FIGURE 7. Plot of $\vec{w}^*$ (horizontal axis) versus $\vec{x}^*$ where $\vec{w}^*$ minimizes (2), $(\vec{x}^*, \vec{y}^*)$ minimizes (10) with $V(x) = 0.1x^4+x^2$, $W(y) = 0.2y^4+y^2$, $N = 1000$ and $\tau = 0.0001$.

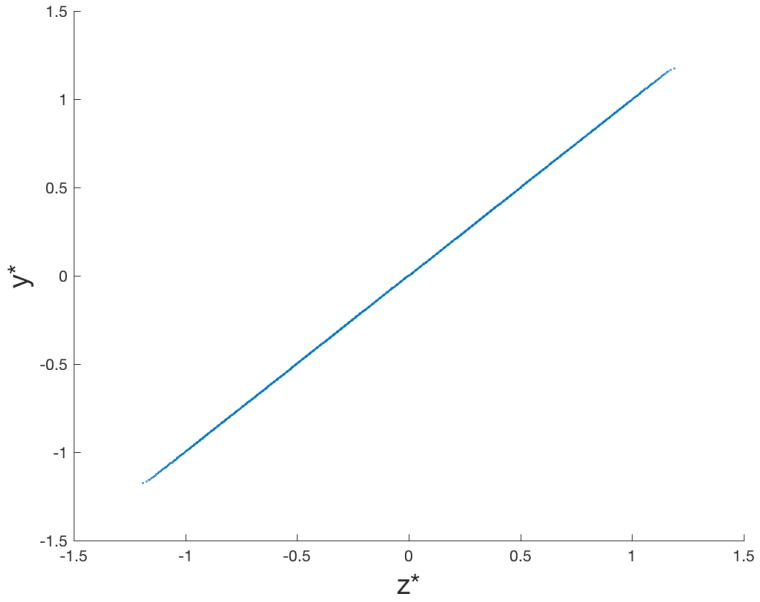

FIGURE 8. Plot of $\vec{z}^*$ (horizontal axis) versus $\vec{y}^*$ where $\vec{z}^*$ minimizes (2) (with $W = V$), $(\vec{x}^*, \vec{y}^*)$ minimizes (10) with $N = 1000$, $V(x) = 0.1x^4 + x^2$, $W(y) = 0.2y^4 + y^2$, and $\tau = 0.0001$.

```
load xmin1000posOnematrix
load ymin1000posOnematrix
load xmin1000posConB
load ymin1000posConB
xtotal=horzcat(-1*fliplr(xmin1000posOnematrix),xmin1000posOnematrix);
ytotal=horzcat(-1*fliplr(ymin1000posOnematrix),ymin1000posOnematrix);
xlength=length(xtotal);
ylength=length(ytotal);
xtotal2=horzcat(-1*fliplr(xmin1000posConB),xmin1000posConB);
ytotal2=horzcat(-1*fliplr(ymin1000posConB),ymin1000posConB);
figure(5)
sz1=1;
scatter(xtotal,xtotal2,sz1)
xlabel('w*','fontsize',18)
ylabel('x*','fontsize',18)
saveas(gcf,'oneandtwo1b.png')
figure(6)
scatter(ytotal,ytotal2,sz1)
xlabel('z*','fontsize',18)
ylabel('y*','fontsize',18)
saveas(gcf,'oneandtwo2b.png')
```

FIGURE 9. The MATLAB code used to create Figures 7 and 8

References

[AGZ10] Greg W. Anderson, Alice Guionnet, and Ofer Zeitouni. *An introduction to random matrices*, volume 118 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2010.

[DE02] Ioana Dumitriu and Alan Edelman. Matrix models for beta ensembles. *J. Math. Phys.*, 43(11):5830–5847, 2002.

[Dei99] P. A. Deift. *Orthogonal polynomials and random matrices: a Riemann-Hilbert approach*, volume 3 of *Courant Lecture Notes in Mathematics*. New York University, Courant Institute of Mathematical Sciences, New York; American Mathematical Society, Providence, RI, 1999.

[Erc11] N. M. Ercolani. Caustics, counting maps and semi-classical asymptotics. *Nonlinearity*, 24(2):481–526, 2011.

[GGPN14] I. P. Goulden, Mathieu Guay-Paquet, and Jonathan Novak. Monotone Hurwitz numbers and the HCIZ integral. *Ann. Math. Blaise Pascal*, 21(1):71–89, 2014.

[GN15] Alice Guionnet and Jonathan Novak. Asymptotics of unitary multimatrix models: the Schwinger-Dyson lattice and topological recursion. *J. Funct. Anal.*, 268(10):2851–2905, 2015.

[HJ90] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, 1990. Corrected reprint of the 1985 original.

[MAT16] MATLAB. *version 9.1.0 (R2016b)*. The MathWorks Inc., Natick, Massachusetts, 2016. `https://www.mathworks.com/help/releases/R2016b/optim/ug/constrained-nonlinear-optimization-algorithms.html?searchHighlight=constrained\%20optimization&s_tid=doc_srchtitle`, Accessed: 2017-11-20.

[Meh81] M. L. Mehta. A method of integration over matrix variables. *Comm. Math. Phys.*, 79(3):327–340, 1981.