# LINEARITY PROBLEMS

TIM

**1. Overview.** Hi Sam, this is a bit to complicate to put in an email.

The issue is that we must be able to harvest the linear map from the transport seep. I have a julia file `LinearQMC.jl` with the codes that generate the problem.

The problem is that linearity fails. We are dead until this gets fixed.

My tests all initialize `qmc_data` with

```
function EasyInit()
    #
    # Init with usual stuff
    #
    Nx = 10     # number of tally cells
    na2 = 11    # number of angles for angular mesh
    s = 1.0     # parameter in Garcia/Siewert
    N = 100    # number of particles per source itertion
    LB = 0.0   # left bound
    RB = 5.0   # right bound
    geometry = "Slab"
    generator = "Sobol"
    qmc_data = garcia_init(geometry, generator, N, LB, RB,
                            Nx, na2, s)
end
```

If I denote the transport sweep map for the flux by $\mathbf{S}$, then in the language of your codes, I compute $\phi_{new}\mathbf{S}(\phi)$ with

```
phiout=qmc_sweep(phi,qmc_data)
phinew=phiout.phi_avg
```

Since I only need the cell-average flux from `qmc_sweep`, I wrote a function that does that.

```
function TSweep(phi, qmc_data)
    phiout = qmc_sweep(phi, qmc_data)
    phinew = phiout.phi_avg
    return phinew
end
```

A transport sweep is the sum of a linear operator applied to the input flux and a constant term that comes from the boundary data and the fixed source.

$$\mathbf{S}(\phi) = \mathbf{M}\phi + \mathbf{b}$$

So, to recover $\mathbf{M}$ all I need to do is compute $\mathbf{b}$, which is

$$\mathbf{b} = \mathbf{S}(\overline{0}) = \mathbf{M}\overline{0} + \mathbf{b}$$

where $\overline{0}$ is the zero vector, and then

$$\mathbf{M}\phi = \mathbf{S}(\phi) - \mathbf{b}$$

In Julia, we get $\mathbf{b}$ from

```
function getb(qmc_data)
    Nx = qmc_data.Nx
    zed = zeros(Nx)
    b = TSweep(zed, qmc_data)
    return b
end
```

So, I can now write the linear operator $\mathbf{M}$ as

```
function MMul(phi, qmc_data, b)
    mmul = TSweep(phi, qmc_data) - b
    return mmul
end
```

**2. Testing Linearity.** I am in great shape if `MMul` is a linear map. This means that for all vectors $\mathbf{x}$ and $\mathbf{y}$ and scalars $\alpha$ and $\beta$

$$\mathbf{M}(a\mathbf{x} + b\mathbf{y}) = a(\mathbf{M}\mathbf{x}) + b(\mathbf{M}\mathbf{y})$$

So I wrote a test to check

```
function test1()
    qmc_data = EasyInit()
    b = getb(qmc_data)
    Nx = qmc_data.Nx
    linerror = 0.0
    for itest = 1:100
        x = rand(Nx)
        y = rand(Nx)
        alpha = rand()
        beta = rand()
        z = alpha * x + beta * y
        mz = MMul(z, qmc_data, b)
        mx = MMul(x, qmc_data, b)
        my = MMul(y, qmc_data, b)
        mxy = alpha * mx + beta * my
        linerror += norm(mz - mxy)
    end
    return linerror
end
```

When I run it I get

```
julia> test1()
3.10686e-14
```

So things look good. But GMRES was breaking because the approximate residuals were far from the real ones. The example I send you and Ryan was the clue. I will test for linearity again with the two vectors from that example.

The problem from yesterday started with two vectors

$$\phi_0 = (1, 1, \ldots, 1)^T$$

and

$$\mathbf{r}_0 = \mathbf{S}(\phi_0) - \phi_0 = \mathbf{b} + \mathbf{M}\phi_0 - \phi_0.$$

We got

$$\mathbf{S}(\mathbf{r}_0 / \|\mathbf{r}_0\|) = \mathbf{S}(\overline{0})$$

which implies that $\mathbf{M}\mathbf{r}_0 = \overline{0}$. I verify that in `test2`, which is pretty much what I send yesterday.

```
function test2()
    qmc_data = EasyInit()
    b = getb(qmc_data)
    Nx = qmc_data.Nx
    phi0 = ones(Nx)
    phi1 = TSweep(phi0, qmc_data)
    r0 = phi1 - phi0
    mr0 = MMul(r0, qmc_data, b)
    # mr0=0. That's the problem I sent yesterday.
    println("M r0 = 0!! Here's the norm ", norm(mr0))
end
```

Yup! Same result.

```
julia> test2()
M r0 = 0!! Here's the norm 0.00000e+00
```

So, let's check linearity one more time. Is

$$\mathbf{M}(\phi_0 + \mathbf{r}_0) = \mathbf{M}(\phi_0) + \mathbf{M}(\mathbf{r}_0)?$$

No.

```
function Linear_QMC()
    qmc_data = EasyInit()
    b = getb(qmc_data)
    Nx = qmc_data.Nx
    phi0 = ones(Nx)
    phi1 = TSweep(phi0, qmc_data)
    r0 = phi1 - phi0
    #
    # Linearity?
    #
    v1 = phi0 + r0
    mv1 = MMul(v1, qmc_data, b)
    mphi0 = MMul(phi0, qmc_data, b)
    mr0 = MMul(r0, qmc_data, b) # this is zero!
    linerror = mv1 - (mphi0 + mr0) # supposed to be zero
    println("This ", norm(linerror), " is supposed to be zero.")
end
```

Here's the trouble. Even though `test1` looked good. The map is not linear.

```
julia> Linear_QMC()
This 2.54114e-01 is supposed to be zero.
```

Help!