ANDERSON AND MCSA

C. T. KELLEY

1. What I did. The idea is to treat a single MCSA iteration as a fixed point map and use that in an Anderson iteration. The fixed point map is `mcsa(x)` where, using the notation from [1] with $A = I - H$, the function is

---

`mcsa`(x)
  $x^{1/2} = Hx + b$
  $r = b - (I - H)x^{1/2}$
  Solve $\hat{A}\delta = r$
  $mcsa(x) = x^{1/2} + \delta$

I'm fixing the length of the walks and not terminating on small weights. Stuart suggested that I look at taking more than one Richardson iteration. I did that and the revised fixed point map is

---

`mcsa`(x)
  $x^{1/2} = x$
  for $i = 1 : N_R$ **do**
    $x^{1/2} = Hx^{1/2} + b$
  end for
  $r = b - (I - H)x^{1/2}$
  Solve $\hat{A}\delta = r$
  $mcsa(x) = x^{1/2} + \delta$

The parameters I can play with are

- the number of samples N_S,
- the length of the walks L_W,
- $N_R \geq 1$ the number of Richardson iterations in `mcsa`, and
- the depth of the Anderson acceleration $m \geq 0$ in Anderson(m).

Remember that Anderson(0) is the same as the MCSA iteration.

2. This is almost a linear problem. Strictly speaking, $x =$ `mcsa`(x) is not a linear problem because of the way the initial distribution is built when you solve $\hat{A}\delta = r$. However, the difference between $x =$ `mcsa`(x) is only the Monte Carlo error. If you replace δ with

$$\delta_{exact} = \sum_{l=0}^{L_W} H^l r \tag{2.1}$$

then $x =$ `mcsa`(x) is an approximation of a linear fixed point problem. I'll demonstrate this by going through the steps in the algorithm. $x^{1/2}$ is the results of N_R Richardsons starting

with x, so

$$x^{1/2} = H^{N_R} x + \sum_{p=0}^{N_R-1} H^p b.$$

So

$$r = b - (I-H)x^{1/2} = b - (I-H)\sum_{p=0}^{N_R-1} H^p b - (I-H)H^{N_R} x.$$

Note that

$$\begin{aligned} I - (I-H)\textstyle\sum_{p=0}^{N_R-1} H^p &= I - \textstyle\sum_{p=0}^{N_R-1} H^p + H\sum_{p=0}^{N_R-1} H^p \\ &= -\textstyle\sum_{p=1}^{N_R-1} H^p + \sum_{p=1}^{N_R} H^p = H^{N_R}. \end{aligned}$$

So

$$r = H^{N_R}(b - (I-H)x).$$

This means the "exact" fixed point map is

$$\texttt{mcsa}_{exact}(x) = \sum_{l=0}^{L_W} H^l r = \sum_{l=0}^{L_W} H^{l+N_R}(b - (I-H)x).$$

The iteration matrix for this map is

$$M = -(I-H)\sum_{l=0}^{L_W} H^{l+N_R}.$$

I'm putting this in here to show why I don't think one needs to redo the analysis of the MC approximation if we take more than one Richardson per MCSA step. The analysis of the correction stays the same. The only thing that looks different is what one wraps around it.

3. Results. All of these experiments use the JPWH_991 example from the Matrix Market. The timings and iteration counts are very strongly correlated.

I did not use any explicit parallelism in Matlab (*i.e.* no `parfor` loops).

All the examples have walks of length $L_W = 10$. I found that using more Richardon's reduced the iteration count by quite a bit. I used $m = 0, 1, 2, 5$, and 10 for Anderson.

3.1. $N_S = 1000$**.** I had to take a few Richardson's to get this to work consistently. The timings for a computation this short are not very illuminating, but do show that fewer iterations lead to faster times.

Here's typical example.

If you increase the number of Richardsons to $N_R = 5$ the iteration count decreases.

TABLE 3.1
Timings: $N_S = 1000$

$N_R = 3$	$N_R = 5$
14.08	12.55
13.98	10.77
5.99	4.41
5.06	4.32
5.80	5.04

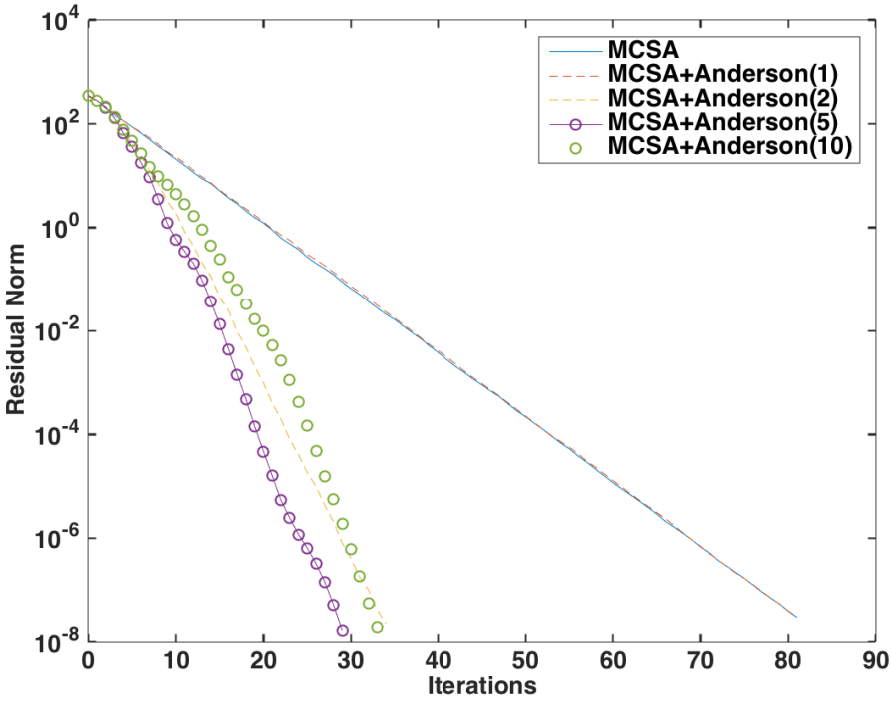

FIG. 3.1. $N_R = 3$, $L_W = 10$, $N_S = 1000$

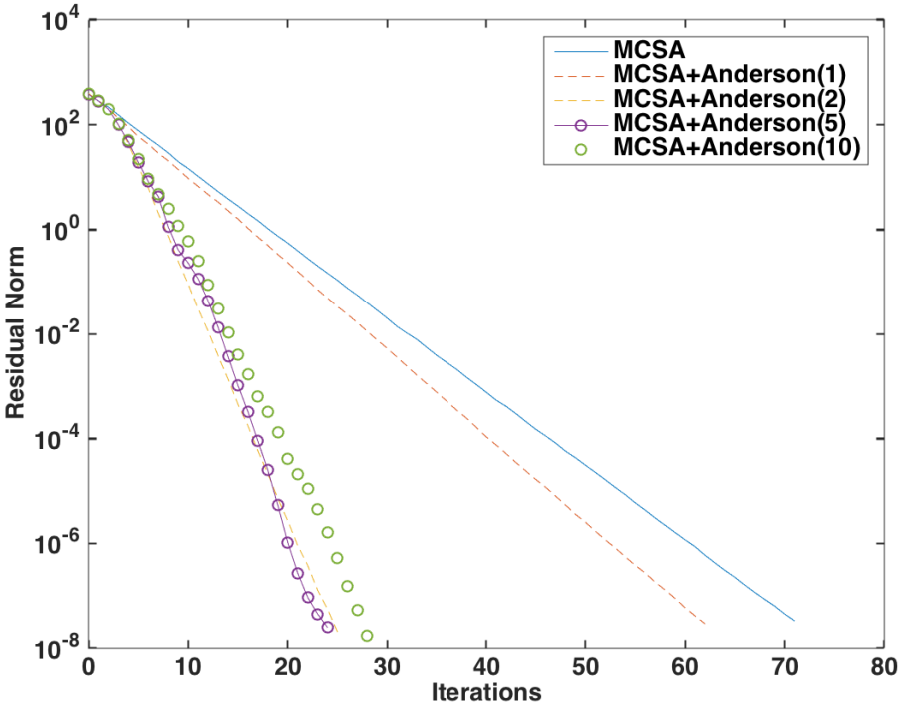

FIG. 3.2. $N_R = 5$, $L_W = 10$, $N_S = 1000$

4. $N_S = 10000$**.** Here one could use one Richardson with no ill effects, but more is better. The rows in the tables are for Anderson(m) with $m = 0, 1, 2, 5, 10$.

TABLE 4.1
Timings: $N_S = 10000$

$N_R = 1$	$N_R = 3$	$N_R = 10$
162.94	141.26	96.47
68.85	43.46	31.25
46.51	41.40	28.04
44.90	33.01	25.86
46.59	33.94	22.85

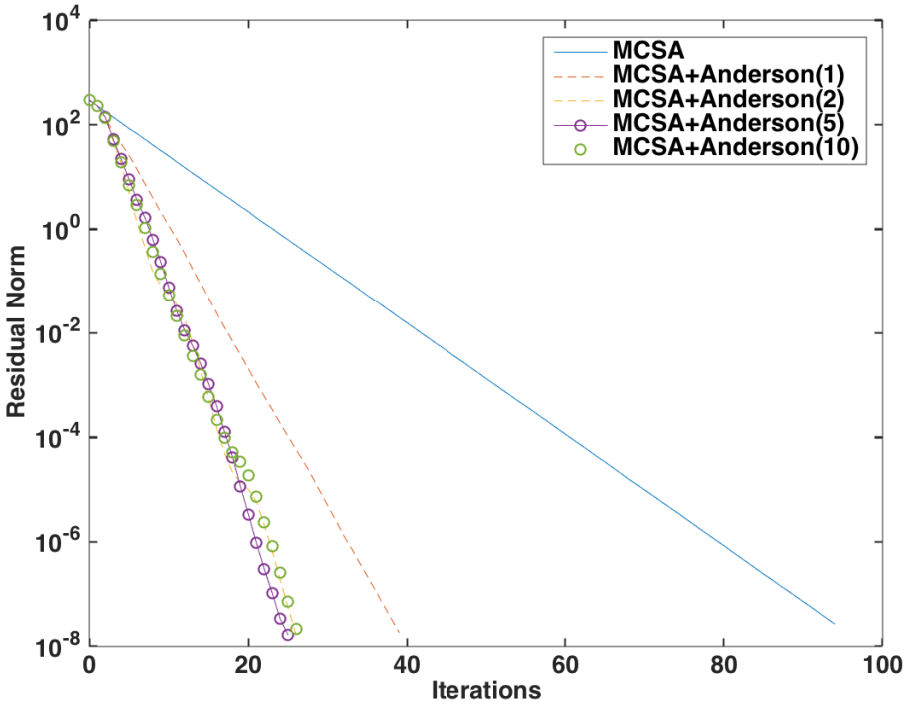

FIG. 4.1. $N_R = 1$, $L_W = 10$, $N_S = 10000$

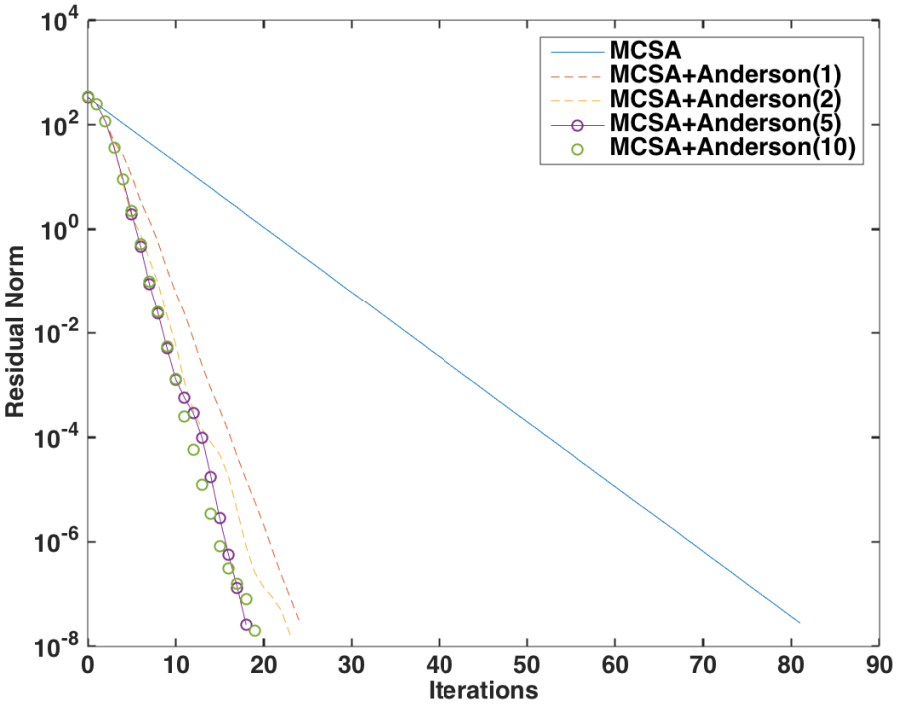

FIG. 4.2. $N_R = 3$, $L_W = 10$, $N_S = 10000$

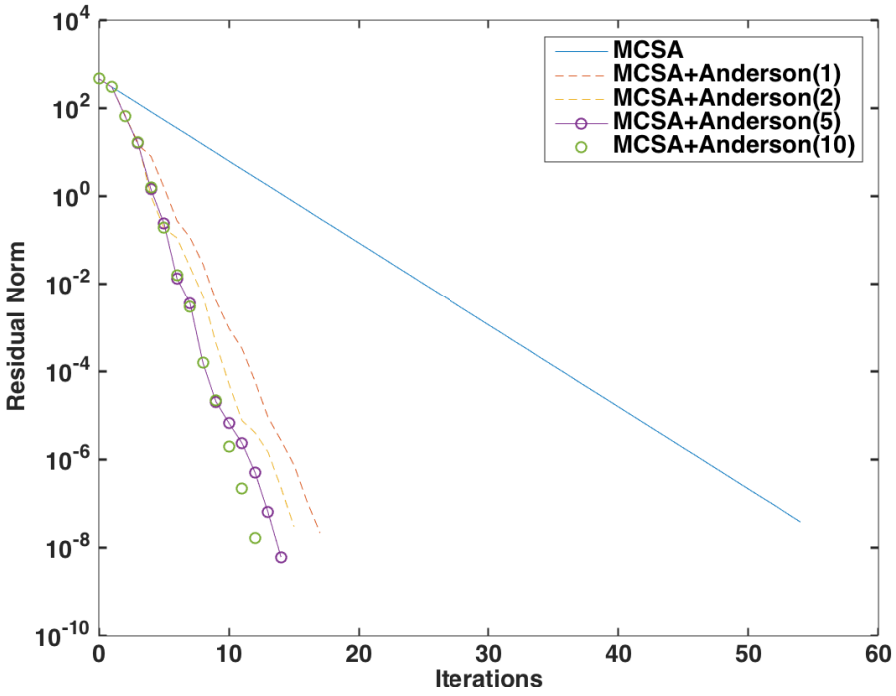

FIG. 4.3. $N_R = 10$, $L_W = 10$, $N_S = 10000$

REFERENCES

[1] T. M. EVANS, S. W. MOSHER, S. R. SLATTERY, AND S. P. HAMILTON, *A Monte Carlo synthetic-acceleration method for solving the thermal radiation diffusion equation*, Journal of Computational Physics, 258 (2014), pp. 338–358.