1 09/27/2016: Convergence of SOR - Linear

This week, I completed a proof that linear SOR converges if the matrix A in the system $A\mathbf{x} = \mathbf{b}$ is positive-semidefinite, if the parameter $\omega \in (0, 2)$, and if in addition to these conditions the block update has a unique minimizer. Similar results have been shown for the Gauss-Seidel algorithm, but those results use the matrix formulation of a full step of the GS algorithm, which precludes easy extension to SOR.

2 10/04/2016: Convergence of SOR - Non-linear

This week, I completed a proof that the Jacobian matrix of a microstep of the non-linear Gauss-Seidel algorithm is identical to that of the linear Gauss-Seidel algorithm. Similar results have been shown for the Gauss-Seidel algorithm, but those results use the matrix formulation of a full step of the GS algorithm, which precludes easy extension to SOR. The result I have shown uses the microstep formulation of the GS algorithm, and is thus trivially extended to SOR.

3 10/11/2016: Convergence of SOR - Cleanup and slides

This week, I have been cleaning up the proof that the Jacobian matrix of a non-linear SOR method is identical to that of linear SOR. The original proof was correct, but somewhat difficult to read due to a lack of good notation for such things as minimization. For clarity, I have moved the justification for some of the more common and less interesting parts of the proof, such as derivation of the matrix form of the Gauss-Seidel microstep from the minimization form.

This cleanup is not yet complete; more time-critical tasks, such as creating the slides for a presentation on the same topic, have taken precedence.

On that note, I have been working on slides for a presentation about the convergence of non-linear SOR when the Hessian matrix of the objective function is positive semi-definite, rather than positive-definite.

4 10/18/2016: Convergence of SOR - Continued

This week, I continued the effort to improve the proof that the Jacobian matrix of a non-linear SOR method is identical to that of linear SOR. I have more carefully stated those things which required further explanation when the proof was discussed. I attempted to re-state the problem in a simpler manner, but doing so introduced additional notation, which ultimately hindered the clarity of the proof.

I have started the process of combining the proofs into a coherent whole. The adapatation of Uschmajew's proof has been updated to correctly reference the proof mentioned above. Redundant sections, including an outdated proof sketch, and the incoherent beginnings of a rejected argument have been trimmed.

5 10/25/2016: Literature

This week, I concentrated on searching the literature for theorems stating the convergence rate of SOR, particularly those which might compare it favorably to that of GS.

Of note, I encountered a theorem, Keller's Theorem, which renders redundant one of the theorems I proved, but only for solving the system $A\mathbf{x} = \mathbf{b}$. More precisely,

- Keller's Theorem indicates that iterations of SOR would converge if applied to the normal equation $A^*A\mathbf{x} = A^*\mathbf{b}$. Because any system may be transformed into a system of this form, it
- If SOR is applied directly to $A\mathbf{x} = \mathbf{b}$, the conclusion of Keller's Theorem does not directly imply that the sequence produced is convergent in the energy seminorm.
- My result shows that the SOR method converges in the energy seminorm when SOR is applied directly to the system $A\mathbf{x} = \mathbf{b}$, again with A being Hermitian positive semi-definite.

The result I found thus has limited utility; it is useful for examining the non-linear SOR algorithm, but is redundant as a method of examining linear SOR.

6 11/01/2016: Examining Conjugate Gradient

This week, I took a closer look at nonlinear Conjugate Gradient. Nonlinear CG is a line-search method which selects as its search direction the direction suggested by linear CG. To be precise, the iterative form of the linear CG method uses information about a previous iteration (in particular, the residual vector) to select a search direction better than the gradient; in nonlinear CG, the residual is replaced by the gradient of the objective function.

In my search so far, I have found global convergence theorems which apply to the regularized tensor optimization problem (indirect minimization of numerical error). I have found no such result for minimization of algebraic error.

At this time, I have not found any statements of the rate of convergence of nonlinear CG. My intuition is that convergence might be as good as quadratic, because linear CG is (theoretically) a direct method[1], and because each step of nonlinear CG is effectively a partial inversion of the Hessian[2].

Though a decade old, the paper found at has been quite helpful, and the newer papers which I have read have not contained significant improvements on the knowledge contained in the older paper.

[1]That is, gives an exact answer in finitely many steps.

[2]A full inversion would give Newton's method, which has quadratic convergence.

7 11/07/2016: Experiments with Conjugate Gradient

This week, I implemented the non-linear Conjugate Gradient algorithm in C++. Several bugs caused weird behavior; these bugs were not related to the actual implementation of the CG algorithm, but instead related to hastily created objective functions. For your amusement, the bugs encountered are stated here:

1. Unexpectedly high iteration counts for quadratic error function. *The objective function used was $\|A\mathbf{x} - \mathbf{b}\|_2^2$, but the gradient used was $\nabla\|\mathbf{x} - A^\dagger\mathbf{b}\|_A^2$.*

2. In separable approximation, error always tends toward the same value. *I forgot to increment an iterator. The end result was that all parameters were assumed to be equal.*

After these bugs were corrected, the algorithm exhibited apparently linear convergence when applied to the T1G1 case. Though the rate of convergence was quite good, at 0.9, ALS requires only a single iteration to solve this particular problem.

Experiments with the T2G2 and T3G2 cases have exhibited both linear convergence, with rates between (0.5 and 0.99), and non-convergence. In one notable case, conjugate gradient required 1044 steps to reach an error of 1.5×10^{-6}, while ALS required only 405 microsteps to reach 1×10^{-14}.

Switching the variant of CG from Fletcher-Reeves to Polak-Ribière provided more consistent convergence, typically fewer than 150 iterations. The convergence remains linear, but this more sophisticated variant of CG appears to beat ALS in all but a few exceptional cases. In some examples, SOR outperforms even this more sophisticated CG method.[3]

I have placed source C++ source code for my implementation of the non-linear CG algorithm on the next page.

[3]There's hope yet!

```
//  Conjugate gradient can have a user-specified gradient...
template<typename F, typename V, typename DelF>
V conjugate_gradient (const F& f, V x, const DelF& gradient)
{
//  Establish the object used to store real numbers.
  typedef typename V::value_type R;
//  Exit once gradient is smaller than this:
  constexpr R kEpsilon = 1.0e-6;

//  As an initial guess, search along the gradient.
  V s = -gradient(x);
//  The direction update needs to know about previous steps.
  auto gradient_length_squared = dot(s, s);
  V delta_ng = -s;

  do {
    R alpha = 0.0;
//  Line search. Store the result in alpha.
    numerical::minimize_1d([&](const R& t)->R {
                                 return f(x + s * t);
                               }, &alpha);
    x = x + s * alpha;

    const V g = gradient(x);
    auto g_len = dot(g, g);
    delta_ng = delta_ng - g;

//  Update search direction using one of several well-known formulae.
    auto beta = -dot(g, delta_ng) / gradient_length_squared; //
Polak-Ribiere

    gradient_length_squared = g_len;
    delta_ng = g;

    s = s * beta - g;
  } while (gradient_length_squared > kEpsilon);
  return x;
}

//  ...Or the gradient can be calculated numerically (not included).
template<typename F, typename V>
V conjugate_gradient (const F& f, const V& x) {
  return conjugate_gradient(f, x, [&](const V& z)->V {
                                    return numerical::gradient(f,z);
                                  });
}
```

8 11/14/2016: Numerical comparisons of CG, ALS, and SOR-ALS

The cost of a full step of nonlinear conjugate gradient is approximately that of a microstep of ALS, ignoring some constant factor. More precisely, the orders[4] are similar.

This week, I examined the number of iterations required for each of CG, ALS, and SOR-ALS (with parameter $\omega = \frac{4}{3}$ to return to a target from a perturbation of the target.

The number of iterations required for Conjugate Gradient to converge to $\|\nabla E^2\| \leq 10^{-8}$ remained approximately within the range 19 - 65. This is notably similar to the number of iterations required for linear CG to converge when applied to a linear system with the same number of parameters. When $\|\nabla E^2\|$ was required to reach 10^{-10}, however, CG required approximately 350 iterations, probably due to numerical error.

ALS and SOR-ALS showed less sensitivity to numerical error, but more variability in the number of iterations. ALS has shown convergence in as few as 19, and as many as 7000 iterations. The average[5] was approximately 300 iterations to reach convergence.

SOR-ALS typically outperformed ALS, as expected, when ALS required a large number of iterations, but occasionally failed to do so when ALS required only a small number of iterations.

In hindsight, this behavior should not be unexpected. Consider the system of linear equations

$$I\mathbf{x} = \mathbf{b}$$

For any $\omega \neq 1$, the rate of convergence of the SOR method will be strictly less than that of the GS method; similarly, perturbations of I will display the same behavior. My intuition is that the systems on which SOR will provide an advantage must be those for which the matrix is ill-conditioned, but this has not yet been verified.

[4]Using big-O notation.
[5]Not formally measured.

9 11/21/2016: SOR-ALS is not new. Convergence results probably are.

This week, I tracked down the paper which mentions "extrapolated ALS", and found that it does indeed suggest the SOR-ALS method. On the bright side, for the purposes of my research, the paper does not provide any convergence results and instead states only a description of the algorithm and a suggestion for the choice of the extrapolation parameter.

`http://www.psychology.uwo.ca/faculty/harshman/wpppfac0.pdf`

10 11/28/2016: Literature and understanding the Hierarchical Tensor format

This week, I searched the literature for further references to the earlier instance of the SOR-ALS method (which previous authors have called HL-PARAFAC). Though it has been mentioned in several papers, I have not yet found any paper which examines it in any detail.

There is a tensor toolkit which might have implemented the algorithm, and indeed does employ some form of linear extrapolation; I will need more time to determine whether the algorithm employed is indeed SOR-ALS.

I also examined the Hierarchical tensor format, seeking to understand the format and any advantages it provides, other than the convergence properties granted by the format. I have concluded that the advantages provided are the aforementioend convergence properties and a potential for reduced space for some tensors, particularly those for which complexity is limited to a few directions (*modes*).

11 12/12/2016: An attempt to show SOR is better than GS

Though we have an example for which SOR does not outperform GS, examination of eigenvalues of the linear SOR update matrix indicates that SOR usually does provide some advantage when used to approximate a positive-definite matrix. This week, I attempted a formal proof that SOR is strictly better than GS, with the hope that this would reveal sufficient conditions under which this occurs. Though I have not yet managed to create such a proof, the process did yield an elegant simplification of the derivative of the SOR update matrix Q_ω. In particular, if the (block) diagonal D is nonsingular, then

$$\frac{\partial Q_\omega}{\partial \omega} \quad = \quad -(D+\omega L)^{-1} D (D+\omega L)^{-1} A \,.$$

I hope to use this to show that $\omega = 1$, which induces the GS algorithm, is suboptimal.