NUEN 629: Homework #4

Thursday, November 19, 2015

Ryan G. McClarren

James B. Tompkins

Problem 1

1. Step Discretization Problem Solution

We are first asked to solve a 1D transport problem highly dependent upon scattering ($Q = 0.01$ and $\Sigma_t = \Sigma_s = 100$). Several methods are involved the solutions given in the listings below.

Listing 1: Diamond Difference Spatial Discretization

```
def dd_sweep1D(I,hx,q,sigma_t,mu,boundary):
  """Compute a transport diamond difference sweep for a given
  Inputs:
    I:              number of zones
    hx:             size of each zone
    q:              source array
    sigma_t:        array of total cross-sections
    mu:             direction to sweep
    boundary:       value of angular flux on the boundary
  Outputs:
    psi:            value of angular flux in each zone
  """
  assert(np.abs(mu) > 1e-10)
  psi = np.zeros(I)
  ihx = 1./hx
  if (mu > 0):
    psi_left = boundary
    for i in range(I):
      psi_right = (q[i]*0.5 + (mu*ihx-0.5*sigma_t[i])*psi_left)\
                  /(0.5*sigma_t[i] + mu*ihx)
      psi[i] = 0.5*(psi_right + psi_left)
      psi_left = psi_right
  else:
    psi_right = boundary
    for i in reversed(range(I)):
      psi_left = (q[i]*0.5+ (-mu*ihx-0.5*sigma_t[i])*psi_right)\
                 /(0.5*sigma_t[i] - mu*ihx)
      psi[i] = 0.5*(psi_right + psi_left)
```

Listing 2: Step Spatial Discretization

```
def step_sweep1D(I,hx,q,sigma_t,mu,boundary):
  """Compute a transport step sweep for a given
  Inputs:
    I:              number of zones
    hx:             size of each zone
    q:              source array
    sigma_t:        array of total cross-sections
    mu:             direction to sweep
    boundary:       value of angular flux on the boundary
  Outputs:
    psi:            value of angular flux in each zone
  """
  assert(np.abs(mu) > 1e-10)
  psi = np.zeros(I)
```

Problem 1 continued on next page...

```
  ihx = 1./hx
  if (mu > 0):
    psi_left = boundary
    for i in range(I):
      psi_right = (q[i]/2. + mu*ihx*psi_left)/(mu*ihx + sigma_t[i])
      psi[i] = 0.5*(psi_right + psi_left)
      psi_left = psi_right
  else:
    psi_right = boundary
    for i in reversed(range(I)):
      psi_left = (q[i]/2. - mu*ihx*psi_right)/(sigma_t[i] - mu*ihx)
      psi[i] = 0.5*(psi_right + psi_left)
      psi_right = psi_left
  return psi
```

Diamond difference and step methods are two approaches to spatial discretization of the transport problem to be solved. The code listed is written for 1D problems to perform a sweep to obtain a solution for a given iteration.

Listing 3: Source Iteration Solution Method

```
def source_iteration(I,hx,q,sigma_t,sigma_s,N,BCs,sweep_type,
                     tolerance = 1.0e-8,maxits = 100, LOUD=False ):
  """Perform source iteration for single-group steady state problem
  Inputs:
    I:              number of zones
    hx:             size of each zone
    q:              source array
    sigma_t:        array of total cross-sections
    sigma_s:        array of scattering cross-sections
    N:              number of angles
    BCs:            Boundary conditions for each angle
    sweep_type:     type of 1D sweep to perform solution
    tolerance:      the relative convergence tolerance for the iterations
    maxits:         the maximum number of iterations
    LOUD:           boolean to print out iteration stats
  Outputs:
    x:              value of center of each zone
    phi:            value of scalar flux in each zone
  """
  iterations = []
  Errors = []
  phi = np.zeros(I)
  phi_old = phi.copy()
  converged = False
  MU, W = np.polynomial.legendre.leggauss(N)
  iteration = 1
  while not(converged):
    phi = np.zeros(I)
    #sweep over each direction
    for n in range(N):
      if sweep_type == 'diamond_difference':
        tmp_psi = dd_sweep1D(I,hx,q + phi_old*sigma_s,sigma_t,MU[n],BCs[n])
```

```
      elif sweep_type == 'step':
        tmp_psi = step_sweep1D(I,hx,q + phi_old*sigma_s,sigma_t,MU[n],BCs[n])
      else:
        sys.exit("Sweep method specified not defined in SnMethods")
      phi += tmp_psi*W[n]
    #check convergence
    change = np.linalg.norm(phi-phi_old)/np.linalg.norm(phi)
    iterations.append(iteration)
    Errors.append(change)
    converged = (change < tolerance) or (iteration > maxits)
    if (LOUD>0) or (converged and LOUD<0):
      print("Iteration",iteration,": Relative Change =",change)
    if (iteration > maxits):
      print("Warning: Source Iteration did not converge")
    iteration += 1
    phi_old = phi.copy()
  x = np.linspace(hx/2,I*hx-hx/2,I)
  return x, phi, iterations, Errors
```

Listing 4: GMRES Solution Method

```
import scipy.sparse.linalg as spla
def gmres_solve(I,hx,q,sigma_t,sigma_s,N,BCs, sweep_type,
                tolerance = 1.0e-8,maxits = 100, LOUD=False, restart = 20 ):
  """Solve, via GMRES, a single-group steady state problem
  Inputs:
    I:              number of zones
    hx:             size of each zone
    q:              source array
    sigma_t:        array of total cross-sections
    sigma_s:        array of scattering cross-sections
    N:              number of angles
    BCs:            Boundary conditions for each angle
    sweep_type:     type of 1D sweep to perform solution
    tolerance:      the relative convergence tolerance for the iterations
    maxits:         the maximum number of iterations
    LOUD:           boolean to print out iteration stats
  Outputs:
    x:              value of center of each zone
    phi:            value of scalar flux in each zone
  """
  iterations = []
  Errors = []

  #compute left-hand side
  LHS = np.zeros(I)

  MU, W = np.polynomial.legendre.leggauss(N)
  for n in range(N):
    if sweep_type == 'diamond_difference':
      tmp_psi = dd_sweep1D(I,hx,q,sigma_t,MU[n],BCs[n])
    elif sweep_type == 'step':
      tmp_psi = step_sweep1D(I,hx,q,sigma_t,MU[n],BCs[n])
    #tmp_psi = sweep1D(I,hx,q,sigma_t,MU[n],BCs[n])
```

```
    LHS += tmp_psi*W[n]

  #define linear operator for gmres
  def linop(phi):
    tmp = phi*0
    #sweep over each direction
    for n in range(N):
      if sweep_type == 'diamond_difference':
        tmp_psi = dd_sweep1D(I,hx,phi*sigma_s,sigma_t,MU[n],BCs[n])
      elif sweep_type == 'step':
        tmp_psi = step_sweep1D(I,hx,phi*sigma_s,sigma_t,MU[n],BCs[n])
      #tmp_psi = sweep1D(I,hx,phi*sigma_s,sigma_t,MU[n],BCs[n])
      tmp += tmp_psi*W[n]
    return phi-tmp
  A = spla.LinearOperator((I,I), matvec = linop, dtype='d')

  #define a little function to call when the iteration is called
  iteration = np.zeros(1)
  def callback(rk, iteration=iteration):
    iteration += 1
    if (LOUD>0):
      print("Iteration",iteration[0],"norm of residual",np.linalg.norm(rk))
    iterations.append(iteration[0])
    Errors.append(np.linalg.norm(rk))

  #now call GMRES
  phi,info = spla.gmres(A,LHS,x0=LHS,restart=restart,
                        tol=tolerance,callback=callback)
  if (LOUD):
    print("Finished in",iteration[0],"iterations.")
  if (info >0):
    print("Warning, convergence not achieved")
  x = np.linspace(hx*.5,I*hx-hx*.5,I)
  return x, phi, iterations, Errors
```

Source iteration and GMRES are two solution methods utilized to solve discrete ordinates transport problems.

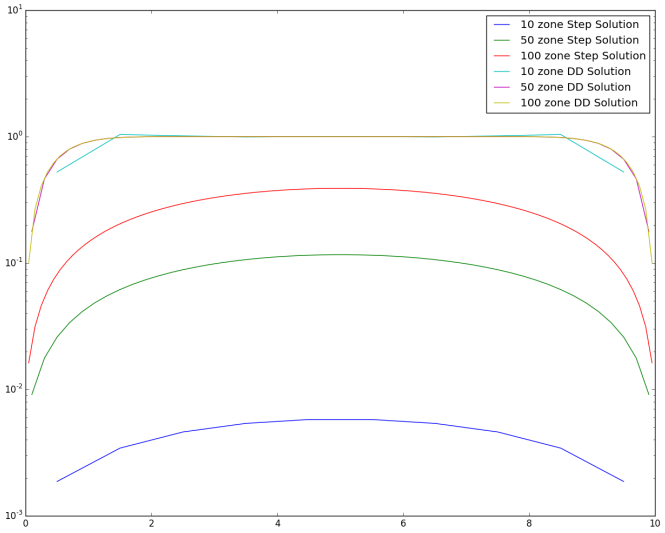

Figure 1: Source Iteration Solutions

Neither spatial discretization method converges for the source iteration solution method for the 10000 maximum iterations specified (excepting the 10 zone step discretization, but it doesn't obtain the correct solution), and in fact, the step solutions seem not to diverge, likely due to the problem being mostly dominated by scattering, thus taking the problem further into the diffusion limit in which the step method does not converge.

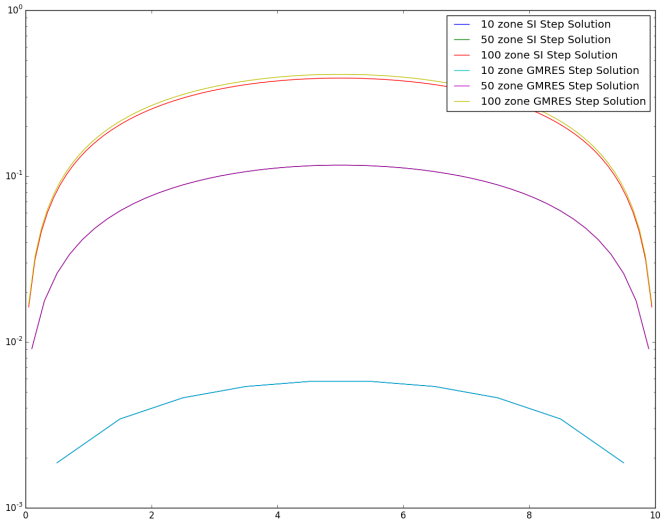

Figure 2: Step Sweep Method Solutions

The GMRES and source iteration method solutions overlap in Figure 2 even with the GMRES solution converging fairly quickly. This demonstrates that regardless of solution method, the step discretization is

not appropriate to use for problems with a lot of scattering.

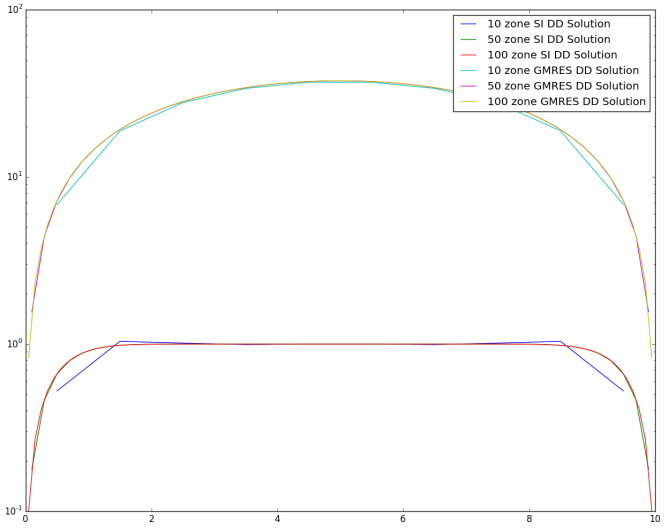

Figure 3: Diamond Difference Method Solutions

The diamond difference solutions are interesting in that it looks like we obtain false convergence with source iteration (the GMRES solutions converge), but as the source iteration method has not converged in any of the spatial resolutions, the solutions end up getting closer to the results obtained with GMRES the more iterations the source iteration solves are allowed to go through.

Problem 1 continued on next page...

2. Difference in Discretizations of Angles

The two forms of the discretizations for step and diamond difference depending upon the angle arise from the direction that the sweep proceeds towards. Each sweep needs a value from the previous zone; so if the sweep is moving in the positive direction originates with the left boundary condition while a sweep in the negative direction requires starting with the right boundary condition. Depending upon the angle from quadrature in which the sweep is moving, one discretization is used over the other seeing as no solution could be obtained starting at the left boundary and moving in a negative direction or at the right moving positive.

3. Error by Iteration Compared to Source Iteration and GMRES

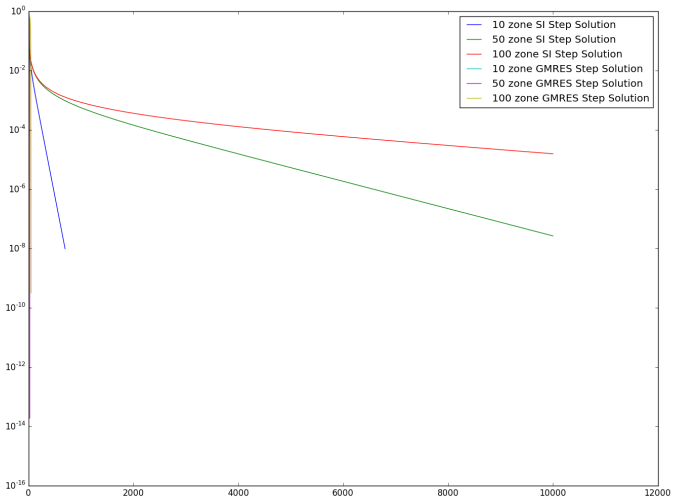

Figure 4: Step Sweep Method Convergence

Plotting the convergence of the step sweep discretization, the decrased convergence rate with increased spatial resolution is obvious. Only the 10 zone case converges, and the flux values are too low to be correct making the method not ideal for the problem.

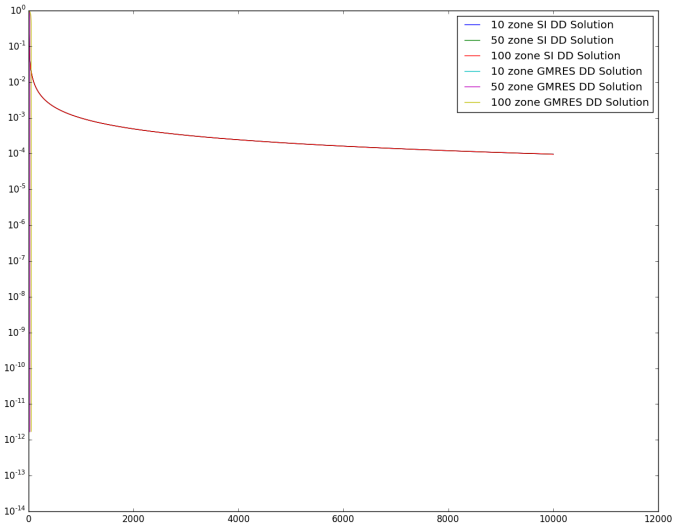

Figure 5: Diamond Difference Method Convergence

All of the source iteration diamond difference solutions converge at similar rates, but still extremely slowly compared to the GMRES method.

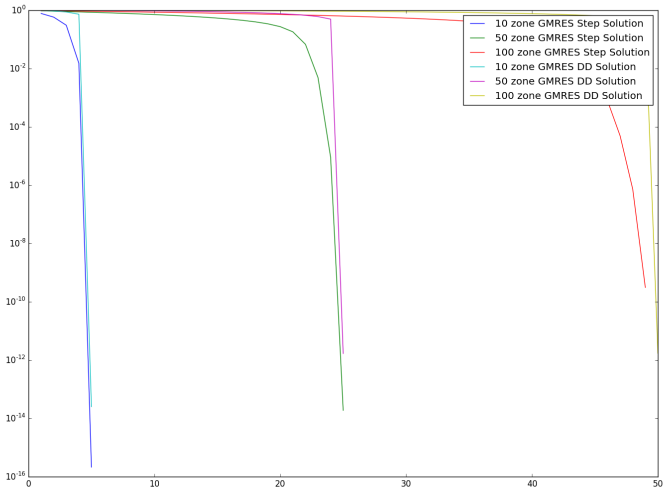

Figure 6: GMRES Convergence

Figure 6 is given to compare the GMRES solutions' convergence. An interesting feature to note for the GMRES diamond difference error is that the rate beings slowly, then in one or two iterations quickly converges to the solution by reducing the error several orders of magnitude. The GMRES step solutions are more gradual, but again, the method is not sound for this problem.

4. Reed's Problem

Reed's problem is a benchmark originally devised to test differencing schemes for discrete ordinates codes. It consists of four mirrored material regions, non-uniform source distribution, and vacuum boundaries (or depending upon how the problem is set up, a reflecting boundary in the middle). Using the methods outlined in the first section of the problem, we are tasked with obtaining and comparing solutions to this problem.

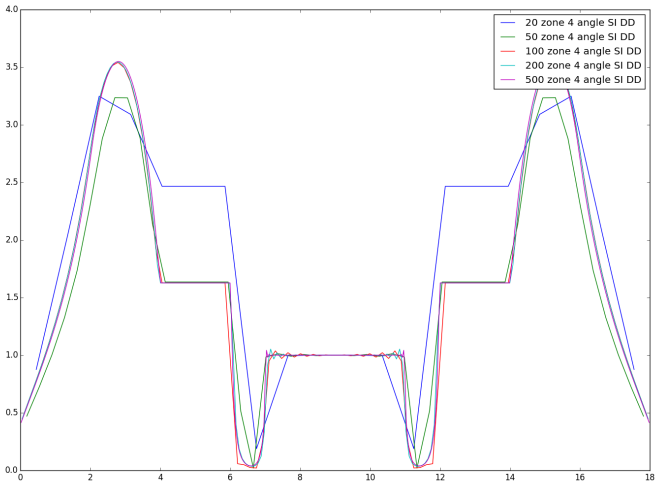

Figure 7: Source Iteration Diamond Difference Solutions with increasing spatial resolution

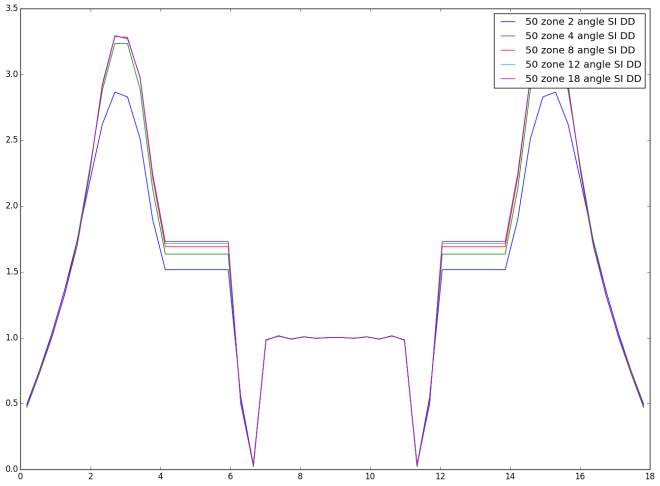

Figure 8: Source Iteration Diamond Difference Solutions with increasing angular resolution

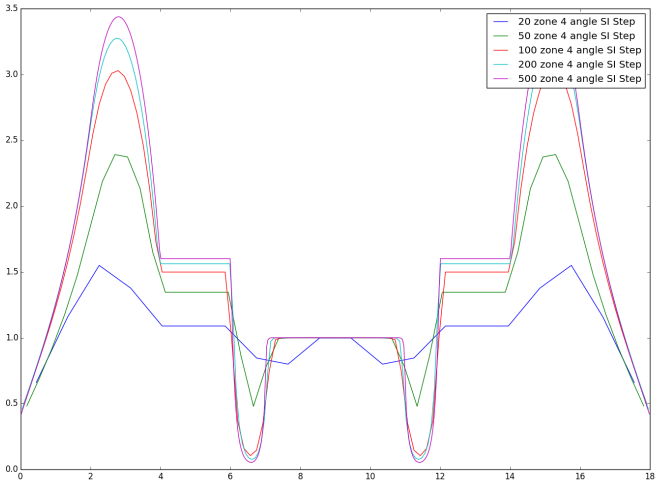

Figure 9: Source Iteration Step Solutions with increasing spatial resolution

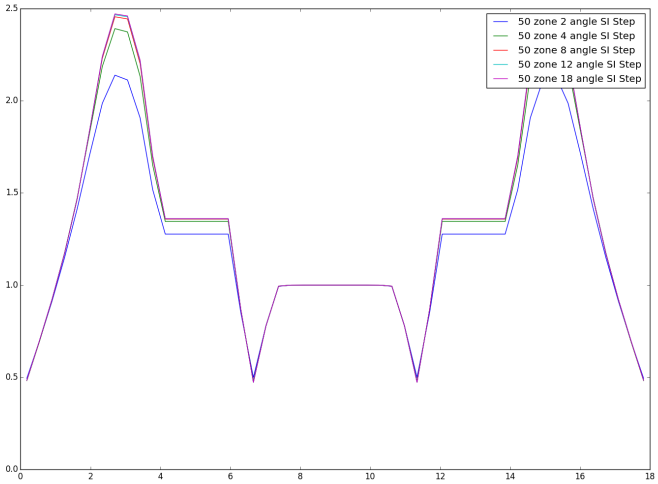

Figure 10: Source Iteration Step Solutions with increasing angular resolution

Problem 1 continued on next page...

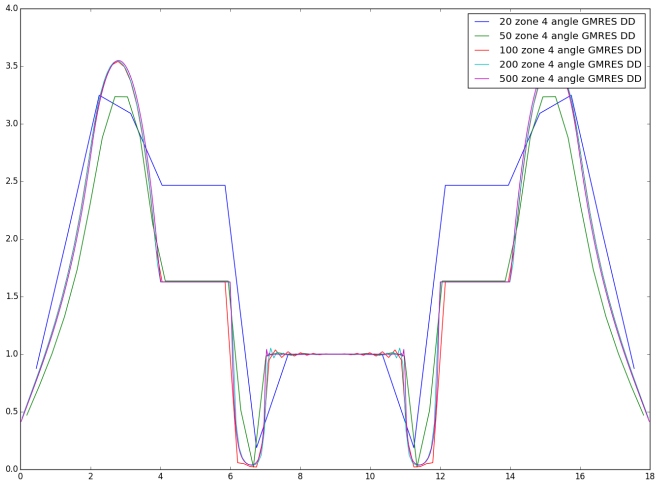

Figure 11: GMRES Diamond Difference Solutions with increasing spatial resolution

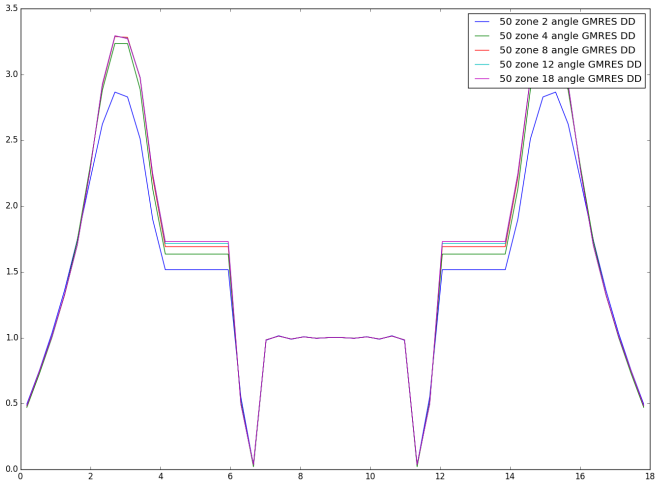

Figure 12: GMRES Diamond Difference Solutions with increasing angular resolution

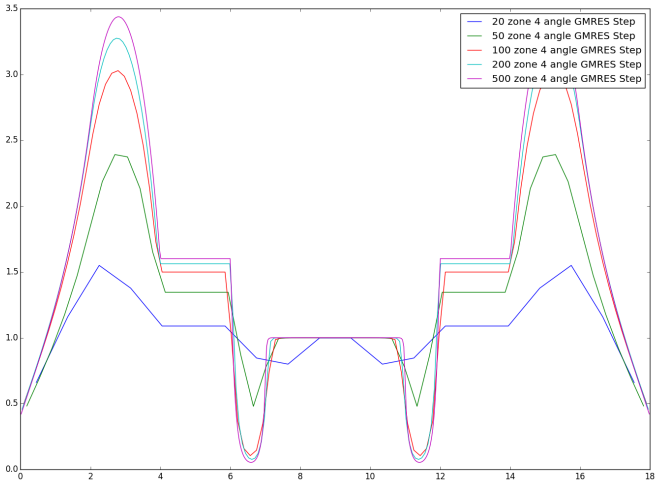

Figure 13: GMRES Step Solutions with increasing spatial resolution

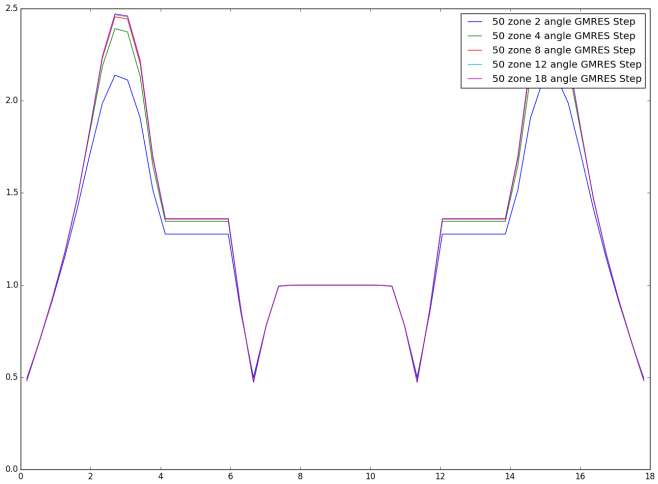

Figure 14: GMRES Step Solutions with increasing angular resolution

Unlike the previous problem solved, all methods converge to a solution with Reed's problem. A few features of the solutions stand out for the various methods when increasing the spatial and angular resolution. The diamond difference method seems to contain spatial oscillations in the center region that increased spatial resolution mitigates. Step solution methods start out as fairly bad approximations of the solution, but quickly converge with increased spatial resolution. It appears that increasing the number of angles in the quadrature scales the solution closer to the true evaluation, but this problem looks more spatially dependent.

Problem 1 continued on next page...

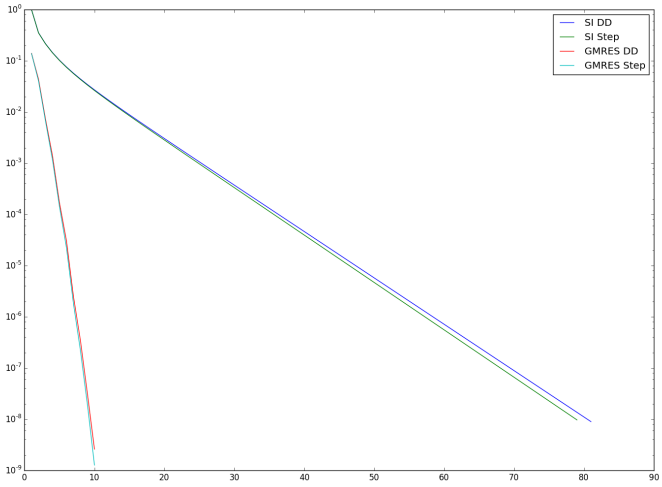

Figure 15: Convergence of solution methods for highest spatial and angular resolution

GMRES converges much faster than source iteration for this problem and the step method converges faster than diamond difference in this problem due to the reduced amount of scattering compared to the previous problem.