Summary of proposed changes to implementation of BiCGStab and CG linear solver methods in AMReX

October 6, 2023

1 BiCGStab method

1.1 BiCGStab: Current implementation

AMReX implements the BiConjugate Gradient Stabilized (BiCGStab) method in MLCGSolverT<MF>::solve_bicgstab. At the time of writing, the current implementation can be summarized as follows:

1. $\hat{p} = \hat{s} = 0$
2. $r = \text{Lp.correctionResidual}(\text{rhs}, \text{sol})$; $r = \text{Lp.normalize}(r)$
3. $\text{sorig} = \text{sol}$; $\hat{r} = r$
4. $\text{sol} = 0$
5. $\text{rnorm} = \text{rnorm0} = \|r\|_\infty$
6. $\rho_1 = \alpha = \omega = 0$
7. For $\text{iter} = 1, 2, \ldots,$ until convergence Do:
8. $\rho = (\hat{r}, r)$
9.   if $\text{iter} = 1$:
10. $p = r$
11.   else:
12. $\beta = \frac{\alpha}{\omega} \frac{\rho}{\rho_1}$
13. $p = r + \beta(p - \omega v)$
14. $\hat{p} = p$
15. $v = \text{Lp.apply}(\hat{p})$; $v = \text{Lp.normalize}(v)$
16. $\text{rhTv} = (\hat{r}, v)$
17. $\alpha = \rho / \text{rhTv}$
18. $\text{sol} = \text{sol} + \alpha \hat{p}$
19. $s = r - \alpha v$
20. $\text{rnorm} = \|s\|_\infty$; check convergence
21. $\hat{s} = s$

22. $t = \text{Lp.apply}(\hat{s})$; $t = \text{Lp.normalize}(t)$
23. $\omega = (t, s)/(t, t)$
24. $\text{sol} = \text{sol} + \omega\hat{s}$
25. $r = s - \omega t$
26. $\text{rnorm} = \|r\|_\infty$; check convergence
27. $\rho_1 = \rho$
28. if convergence reached:
29. $\text{sol} = \text{sol} + \text{sorig}$
30. else:
31. $\text{sol} = \text{sorig}$

where $(\cdot, \cdot)$ represents the inner product, $\| \cdot \|_\infty$ is the infinity norm, $\hat{r}, \hat{p}, \hat{s}$ represent the code variables rh, ph, sh, respectively.

The method $\text{Lp.correctionResidual}(\text{rhs}, \text{sol})$ sets r to $\text{rhs} - L(\text{sol})$, where $L(\cdot)$ is the linear operator. The use of "=" represents a call to some method that changes the MF object such LocalCopy, setVal, LinComb, Saxpy, or Xpay.

1.2 BiCGStab: Proposed implementation

The proposed implementation of the BiCGStab method can be summarized as follows:

1. $r = \text{Lp.correctionResidual}(\text{rhs}, \text{sol})$; $r = \text{Lp.normalize}(r)$
2. $\text{sorig} = \text{sol}$; $\hat{r} = r$
3. $\text{rnorm} = \text{rnorm0} = \|r\|_\infty$
4. $\rho = (\hat{r}, r)$
5. $p = r$
6. For $\text{iter} = 1, 2, \ldots,$ until convergence Do:
7. $\hat{p} = p$
8. $v = \text{Lp.apply}(\hat{p})$; $v = \text{Lp.normalize}(v)$
9. $\text{rhTv} = (\hat{r}, v)$
10. $\alpha = \rho/\text{rhTv}$
11. $\text{sol} = \text{sol} + \alpha\hat{p}$
12. $r = r - \alpha v$
13. $\text{rnorm} = \|r\|_\infty$; check convergence
14. $\hat{s} = r$
15. $t = L(\hat{s})$; $t = \text{Lp.normalize}(t)$
16. $\omega = (t, r)/(t, t)$
17. $\text{sol} = \text{sol} + \omega\hat{s}$

18. $r = r - \omega t$
19. rnorm $= \|r\|_\infty$; check convergence
20. $\rho = (\hat{r}, r)$
21. $\beta = \frac{\rho}{\omega \text{ rhTv}}$
22. $p = r + \beta(p - \omega v)$
23. if convergence reached:
24. do nothing
25. else:
26. sol = sorig

1.3 BiCGStab: Summary

These changes avoid the following:

- calling setVal(RT0.0) on $\hat{p}$, $\hat{s}$, and sol before the iter loop.
- usage of MF s. This reduces memory usage and copying time
- checking if iter $==$ 1 inside the loop
- usage of ρ_1. This reduces the number of arithmetic operations to calculate β
- the LocalAdd operation of sol = sol + sorig at the end if convergence is reached

2 CG method

2.1 CG: Current implementation

AMReX implements the Conjugate Gradient (CG) method in MLCGSolverT<MF>::solve_cg. At the time of writing, the current implementation can be summarized as follows:

1. $p = 0$
2. sorig = sol
3. $r =$ Lp.correctionResidual(sol, rhs)
4. sol = 0
5. rnorm = rnorm0 $= \|r\|_\infty$; check convergence
6. $\rho_1 = 0$
7. For iter $= 1, 2, \ldots,$ until convergence Do:
8. $z = r$
9. $\rho = (z, r)$
10. if iter = 1:
11. $p = z$
12. else:

13. $\beta = \rho/\rho_1$
14. $p = z + \beta p$
15. $q = \text{Lp.apply}(p)$
16. $\text{pw} = (p, q)$
17. $\alpha = \rho/\text{pw}$
18. $\text{sol} = \text{sol} + \alpha p$
19. $r = r - \alpha q$
20. $\text{rnorm} = \|r\|_\infty$; check convergence
21. $\rho_1 = \rho$
22. if convergence reached:
23. $\text{sol} = \text{sol} + \text{sorig}$
24. else:
25. $\text{sol} = 0$
26. $\text{sol} = \text{sol} + \text{sorig}$

2.2 CG: Proposed implementation

1. $\text{sorig} = \text{sol}$
2. $r = \text{Lp.correctionResidual}(\text{sol}, \text{rhs})$
3. $\text{rnorm} = \text{rnorm0} = \|r\|_\infty$; check convergence
4. $\rho = (r, r)$
5. $p = r$
6. For $\text{iter} = 1, 2, \ldots,$ until convergence Do:
7. $q = \text{Lp.apply}(p)$
8. $\text{pw} = (p, q)$
9. $\alpha = \rho/\text{pw}$
10. $\text{sol} = \text{sol} + \alpha p$
11. $r = r - \alpha q$
12. $\text{rnorm} = \|r\|_\infty$; check convergence
13. $\rho_1 = \rho$
14. $\rho = (r, r)$
15. $\beta = \rho/\rho_1$
16. $p = r + \beta p$
17. if convergence reached:
18. do nothing
19. else:
20. $\text{sol} = \text{sorig}$

2.3 CG: Summary

These changes avoid the following:

- calling setVal(RT0.0) on p and sol before the iter loop.
- usage of MF z. This reduces memory usage and copying time.
- checking if iter $==$ 1 inside the loop
- the LocalAdd operation of sol $=$ sol $+$ sorig at the end if convergence is reached