Utilizing Quadruple-Precision Floating Point Arithmetic Operation for the Krylov Subspace Methods

Outline

- Krylov Subspace methods
- Results of Accurate Computing
- Tools for Accurate Computing
- Cost of Quadruple Floating Point Arithmetic
- Conclusion

Krylov Subspace methods

- Series of residual vectors $r_0, r_1, r_2, \ldots, r_k, \ldots$
- Finding a basis: they should be orthogonal
- Converge at most N iterations if its computation is accurate

To test

- What happens to Krylov Subspace Methods in Accurate Computing
- We Compare Convergence History in different Mantissa’s bit for changing Computing Accuracy
- Omni Fortran Compiler is used for this purpose

  http://phase.hpcc.jp/Omni/

Test problem

Toeplitz Matrix, $\mathrm{N} = 200$, $\gamma = 1.7$

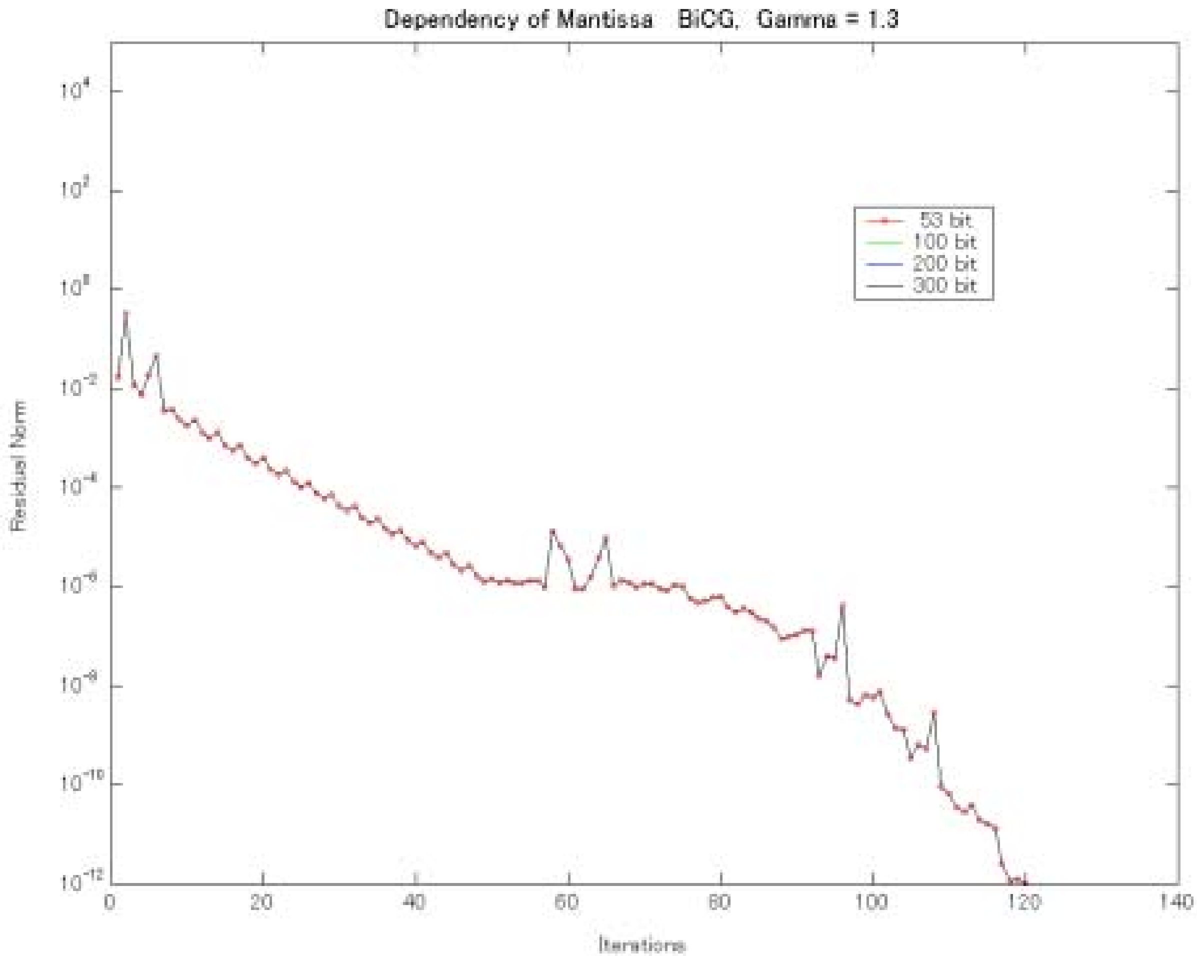
$$A := \begin{bmatrix} 2 & 1 & & & \\ 0 & 2 & 1 & & \\ \gamma & 0 & 2 & 1 & \\ & \gamma & 0 & 2 & \ddots \\ & & \ddots & \ddots & \ddots \end{bmatrix}$$

Right-hand side $b = (1, 1, \cdots, 1)^T$

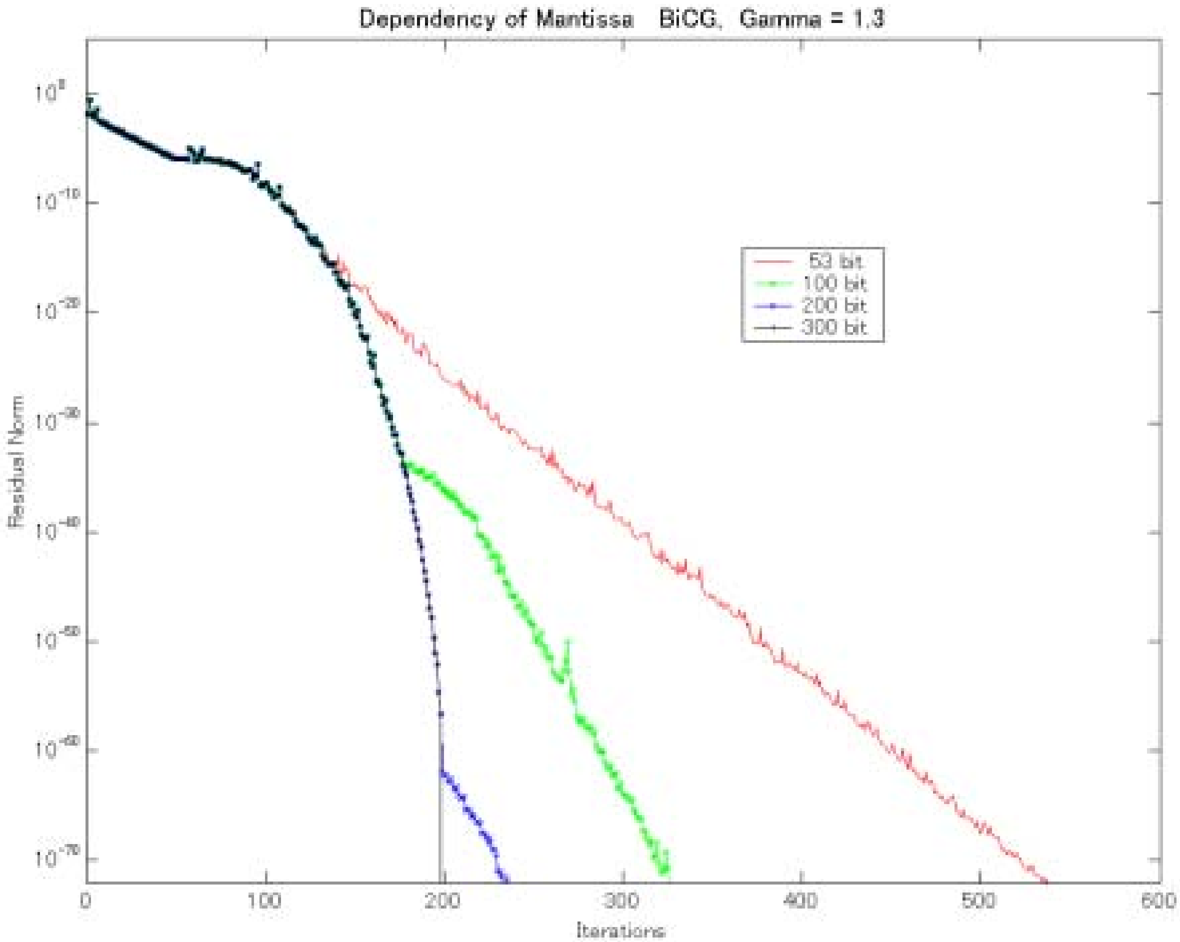
Condition Number

	1.3	1.7	2.1	2.5
1-norm	3.937	10.53	24.93	1936
2-norm	6.463	6.65	16.00	697.
smallest	1.2162 + 1.0105i	1.1044 + 0.7922i	0.8625 + 0.6593i	0.6649 + 0.4963i
largest	4.1091	4.3845	4.6625 + 0.0282i	4.9258 + 0.0336i

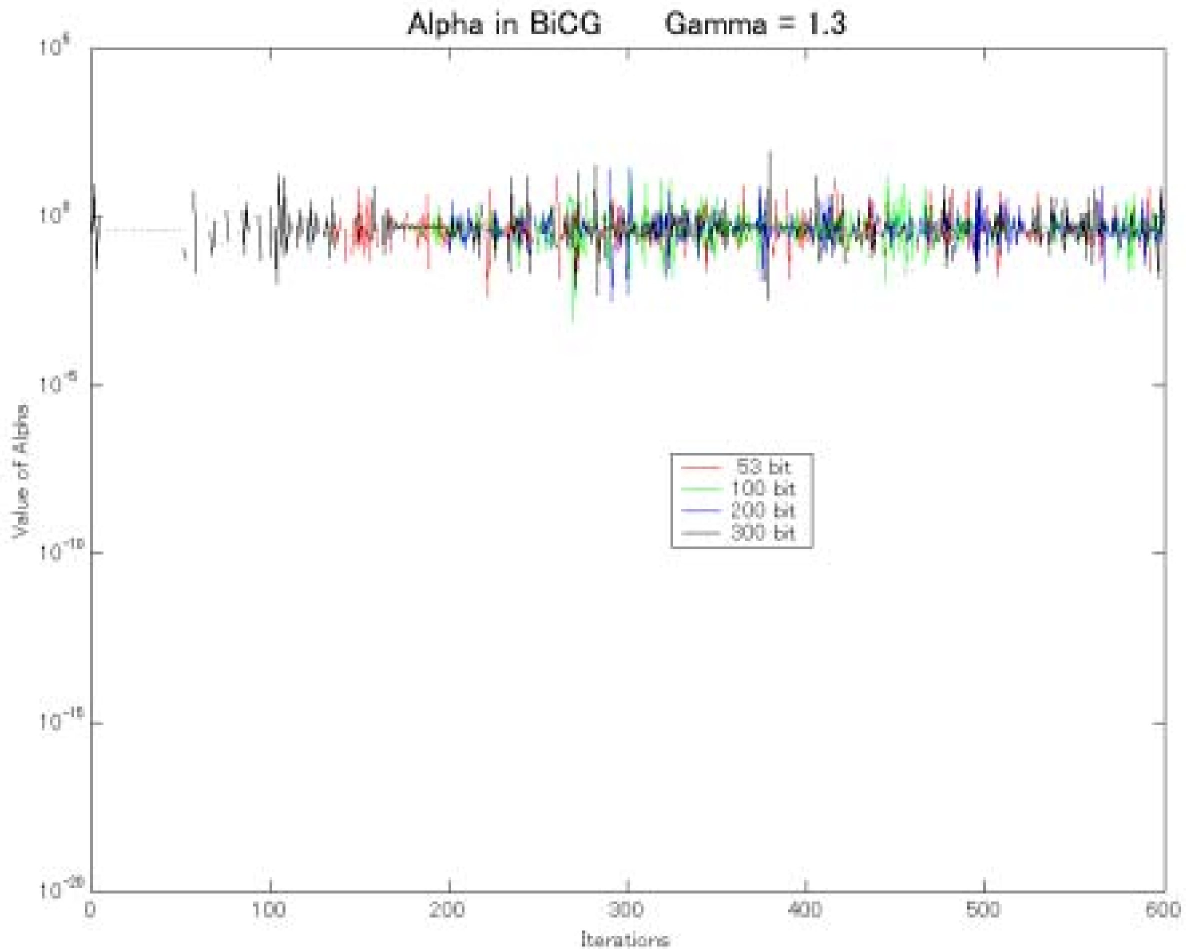
BiCG Gamma = 1.3



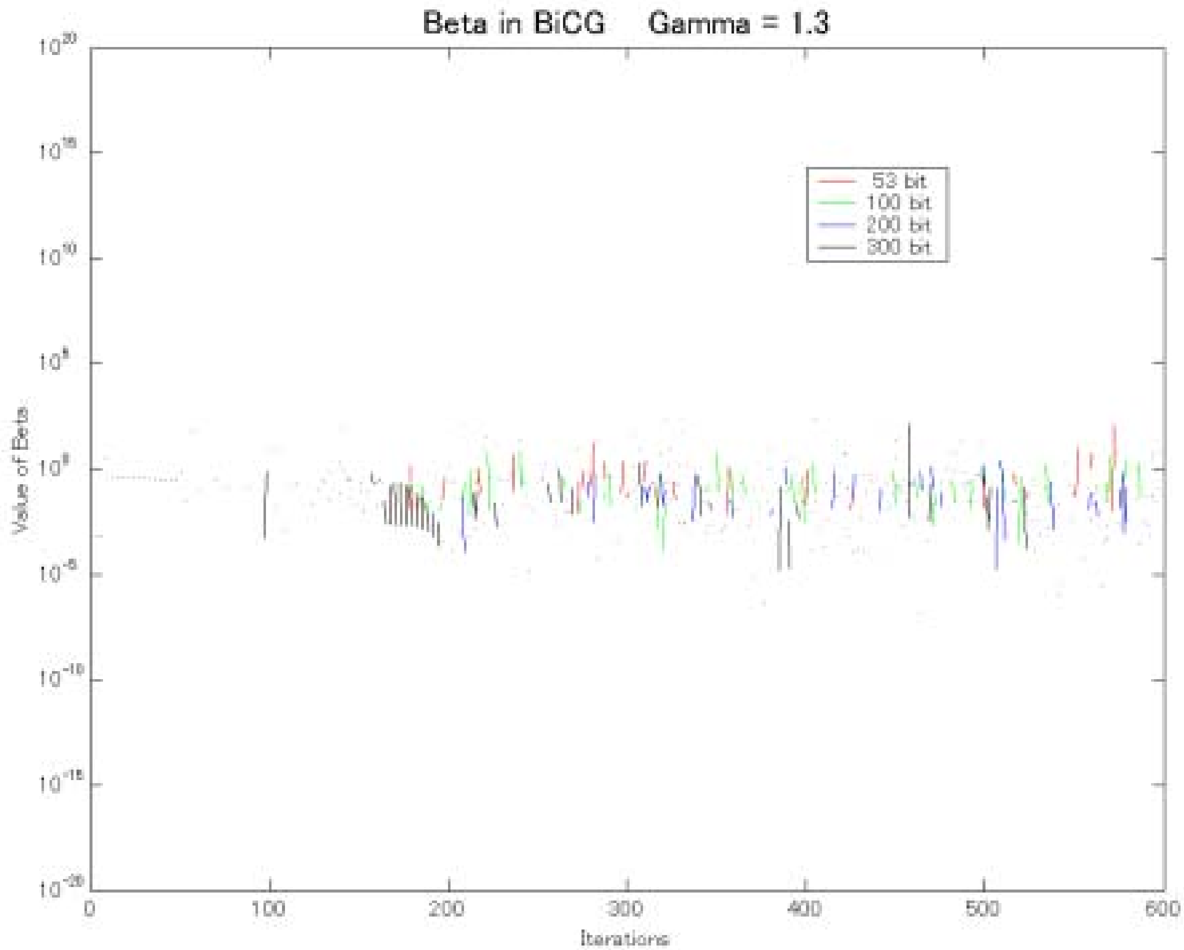
BiCG Gamma = 1.3



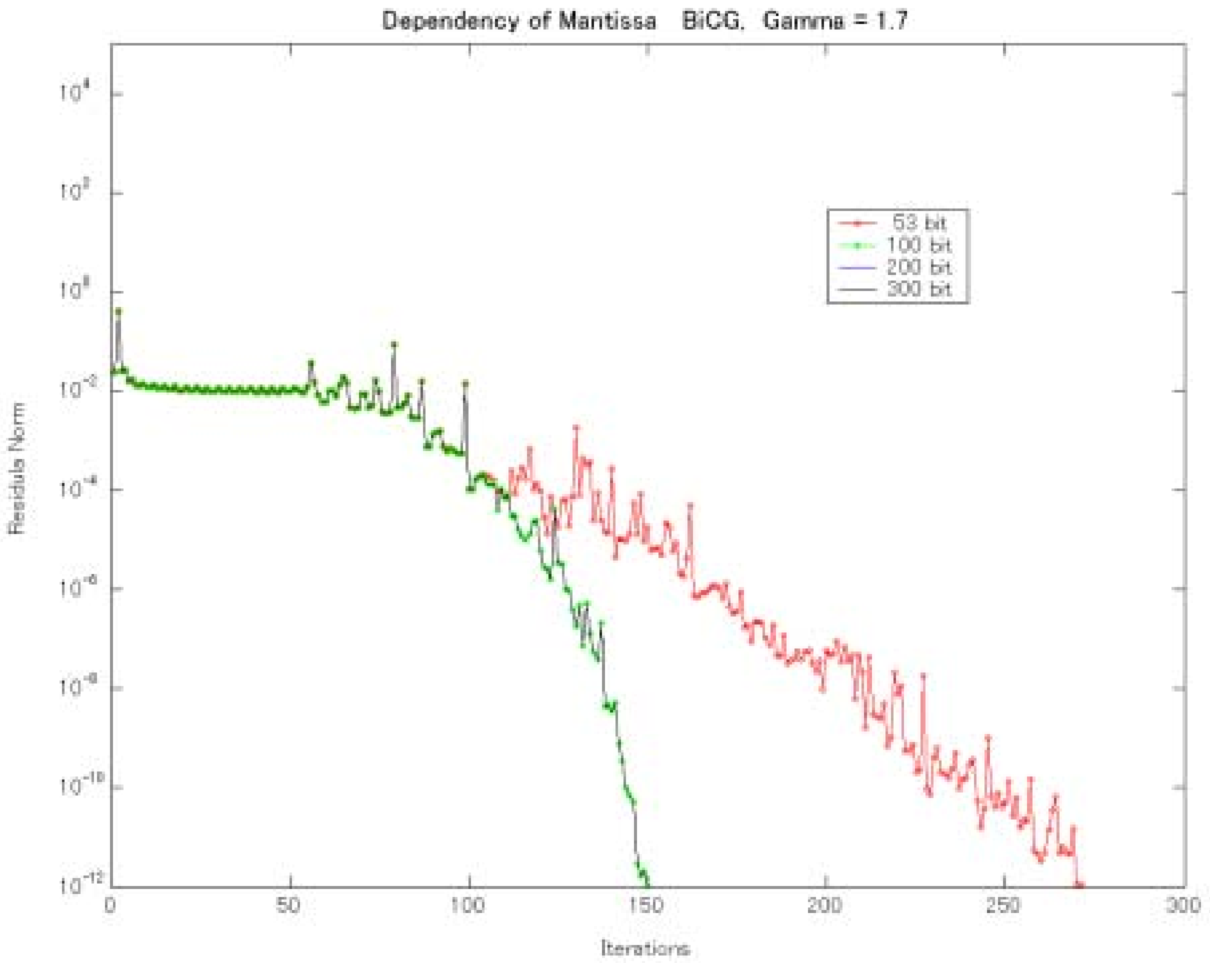
Alpha in BiCG　Gamma = 1.3



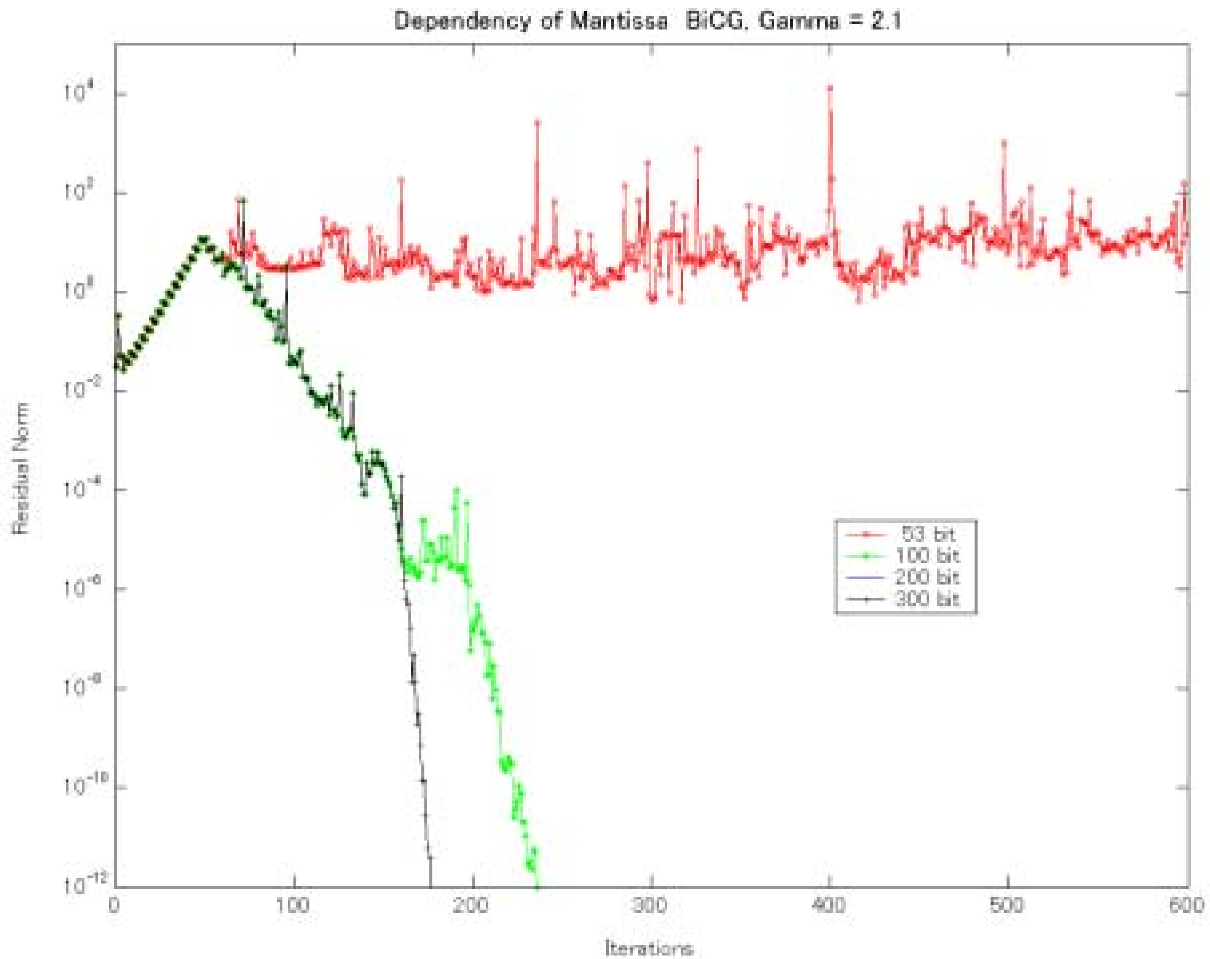
Beta in BiCG Gamma = 1.3



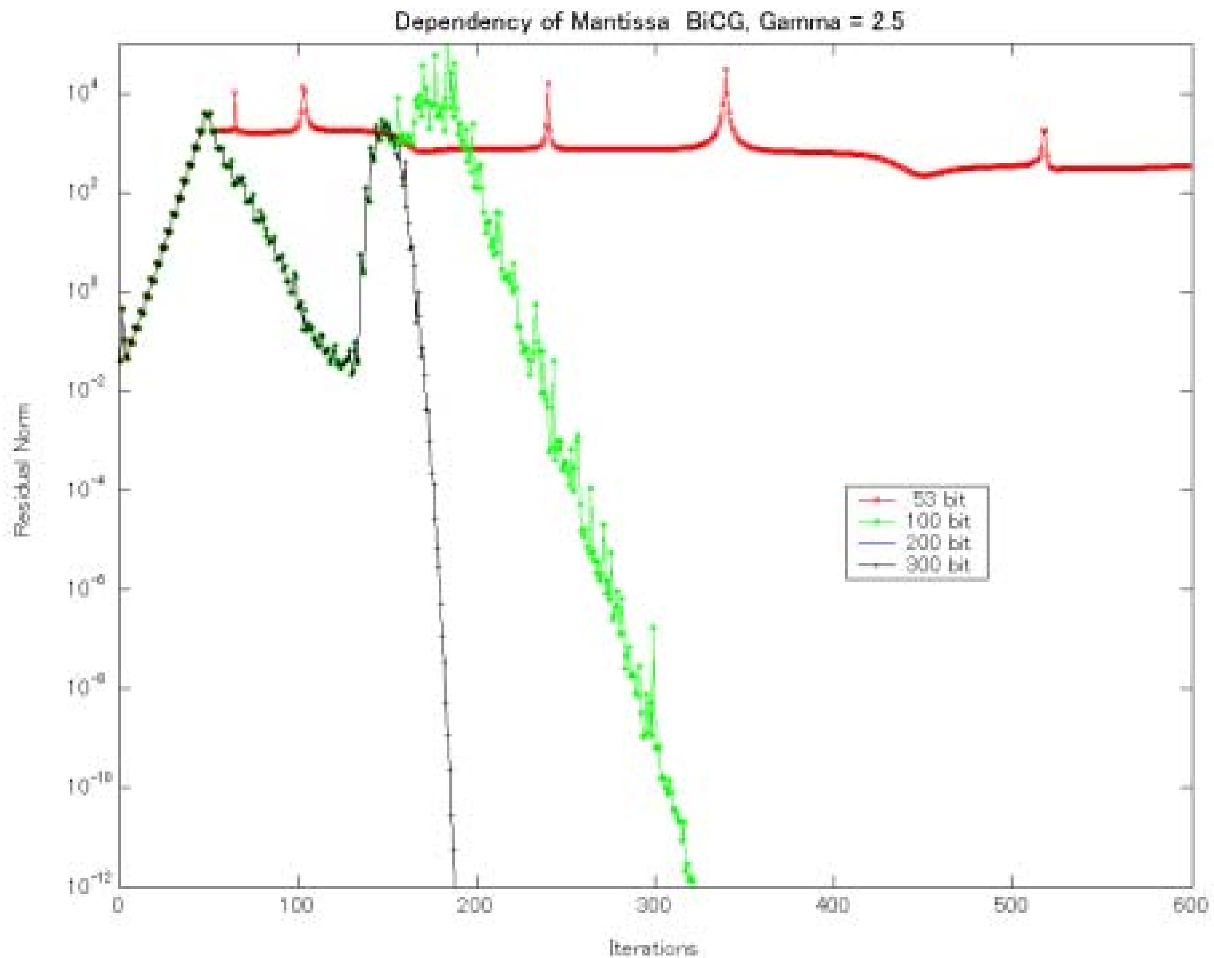
BiCG Gamma = 1.7



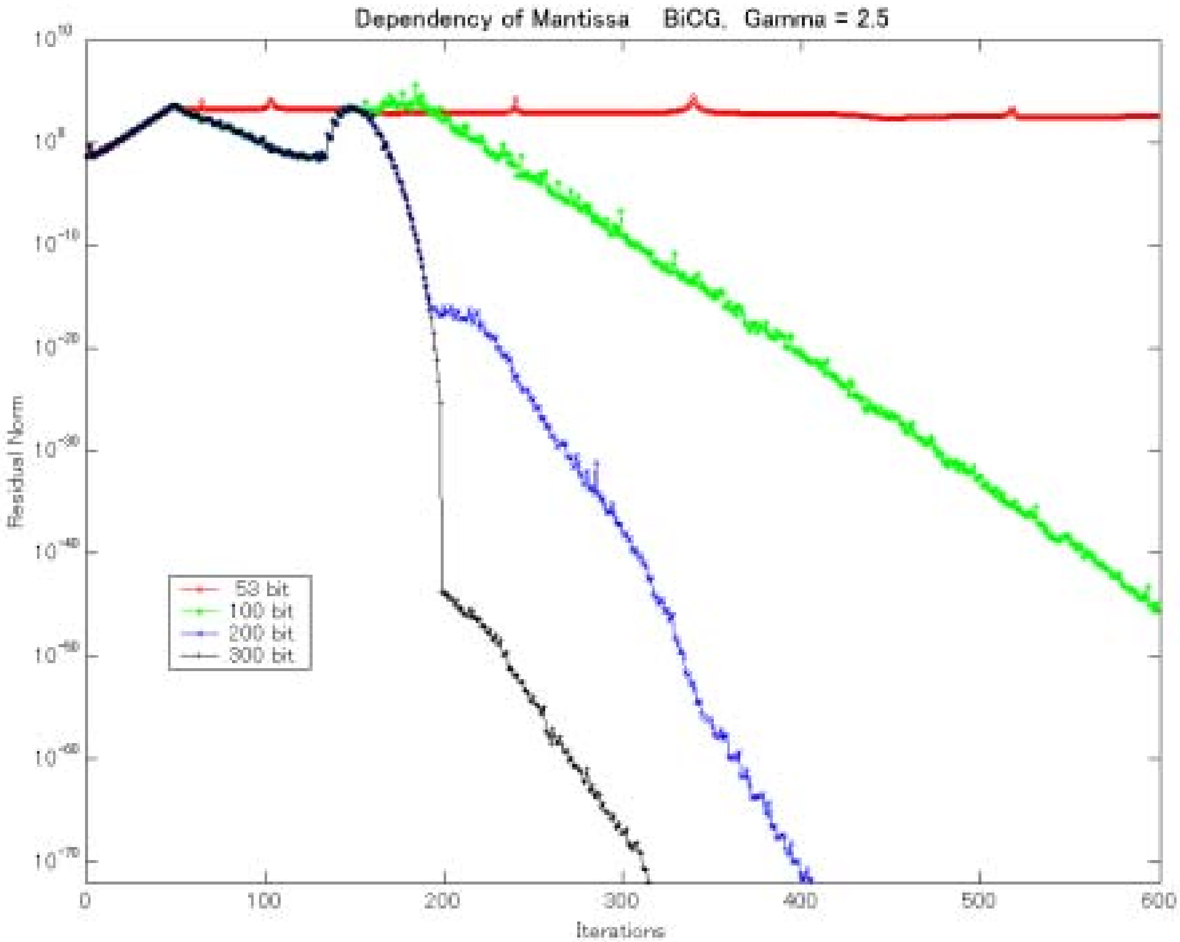
BiCG Gamma = 2.1



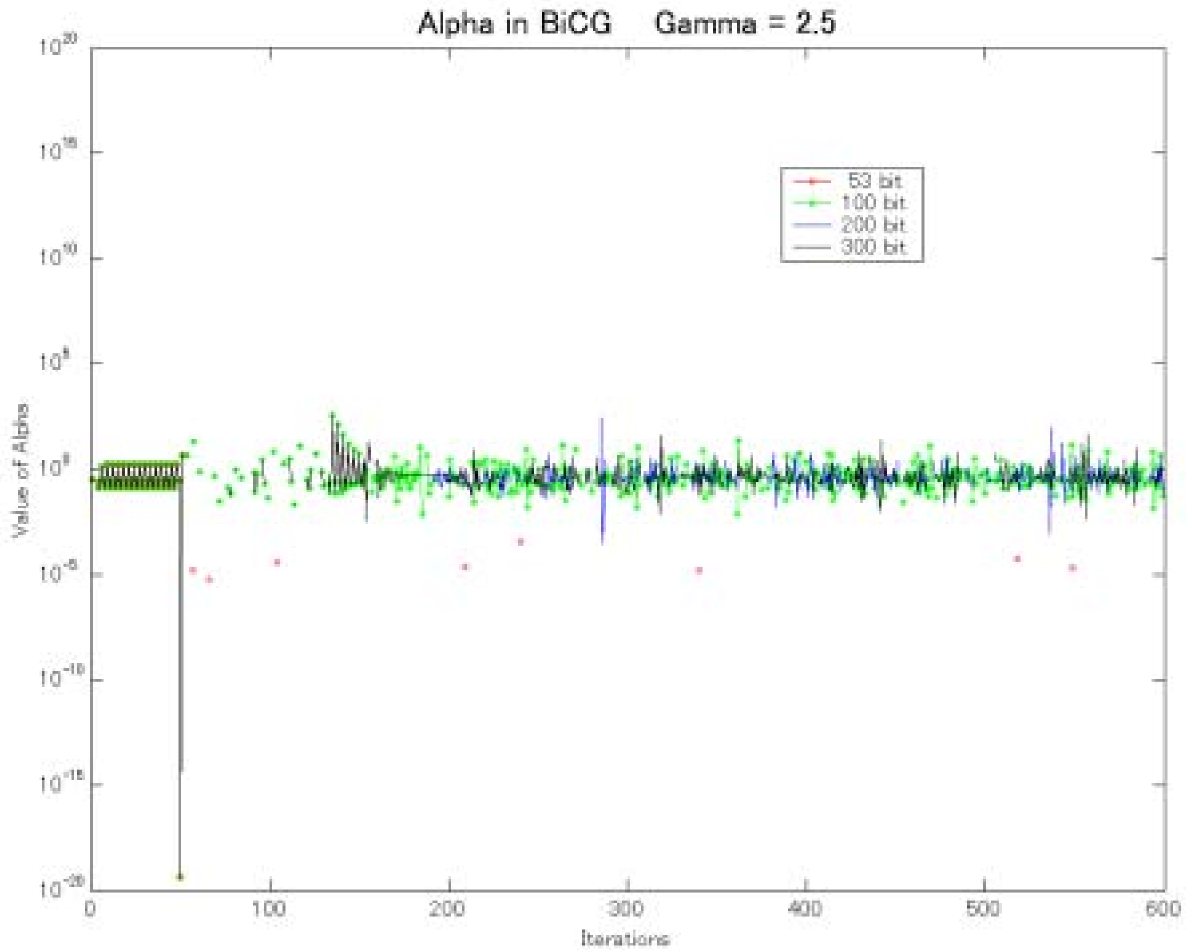
BiCG Gamma = 2.5



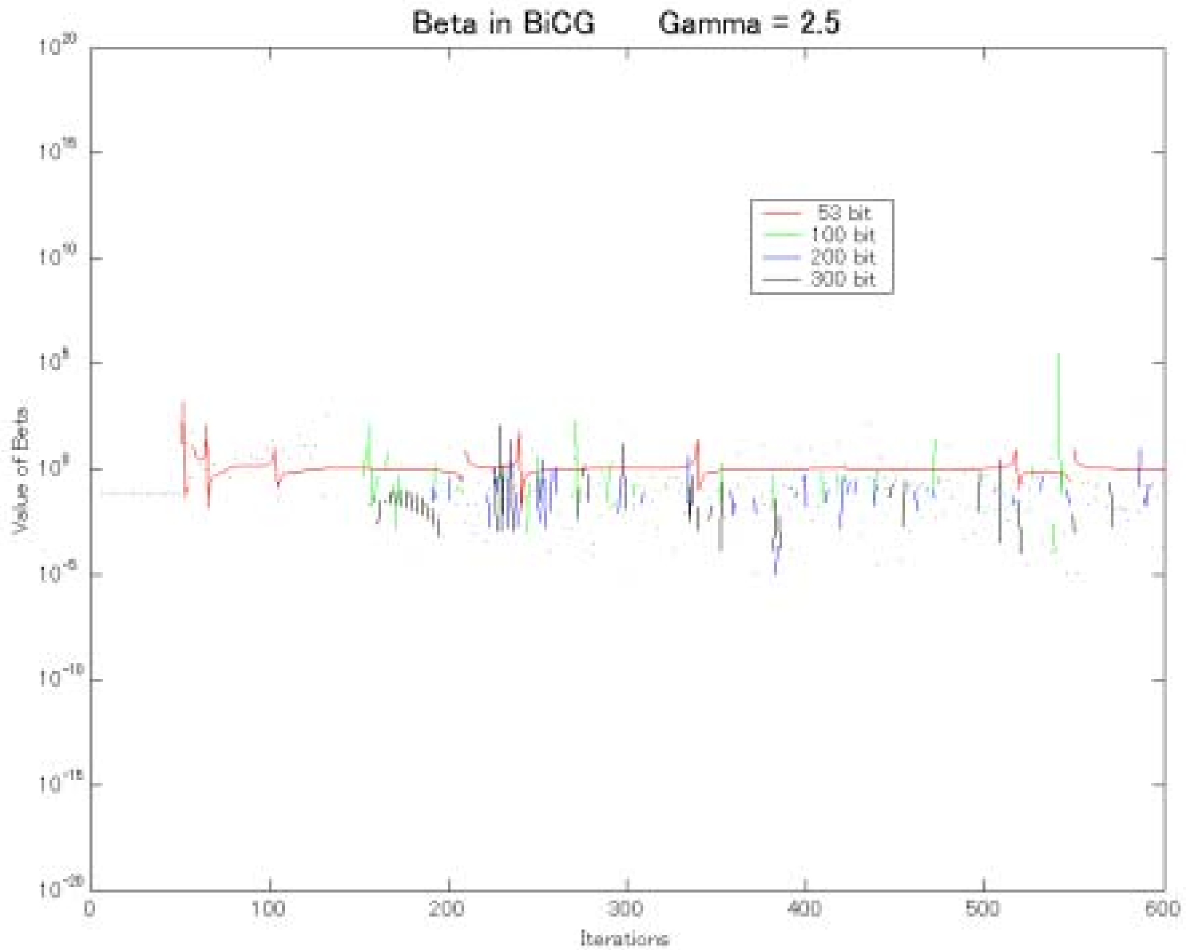
BiCG   Gamma = 2.5



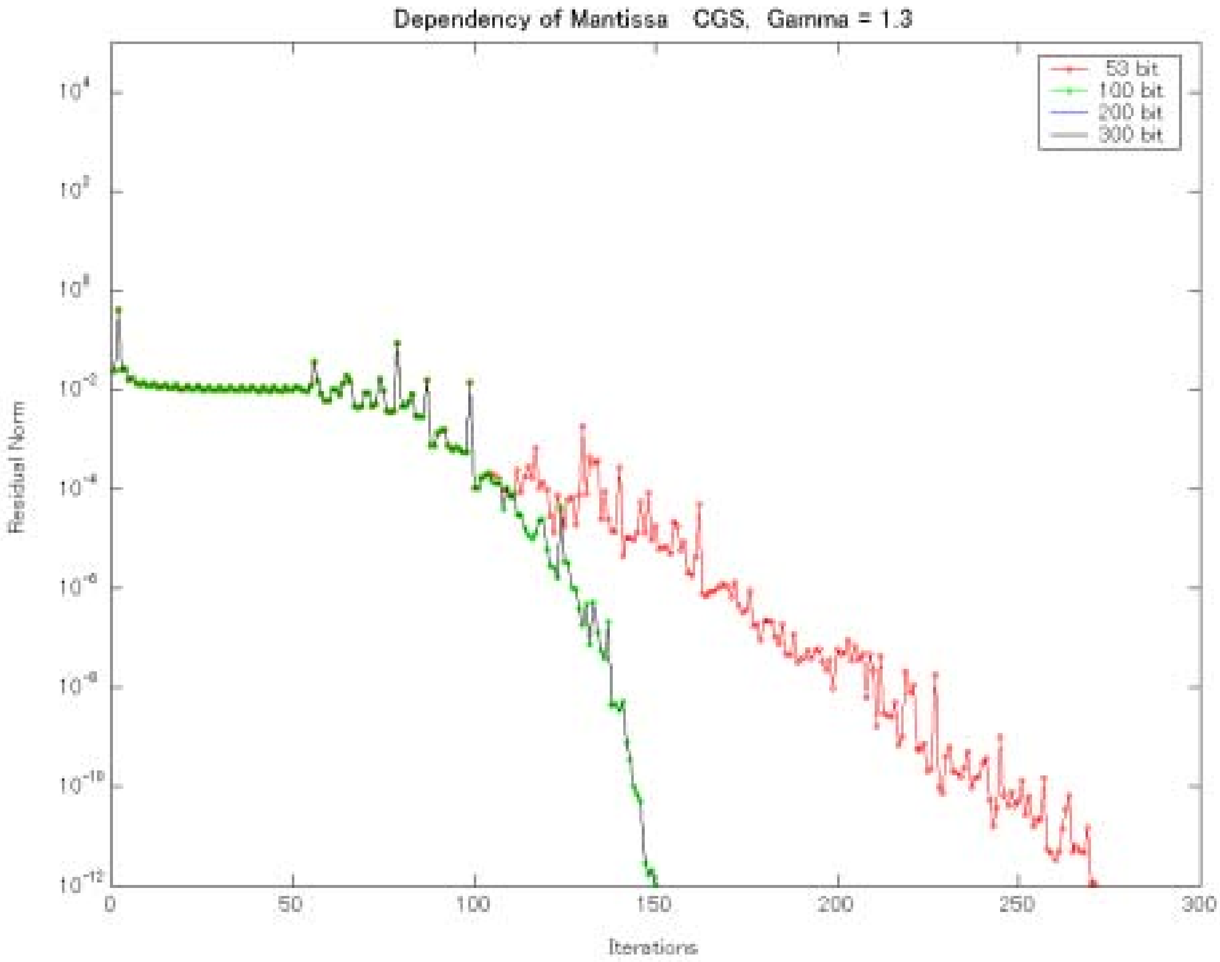
Alpha in BiCG   Gamma = 2.5



Beta in BiCG Gamma = 2.5

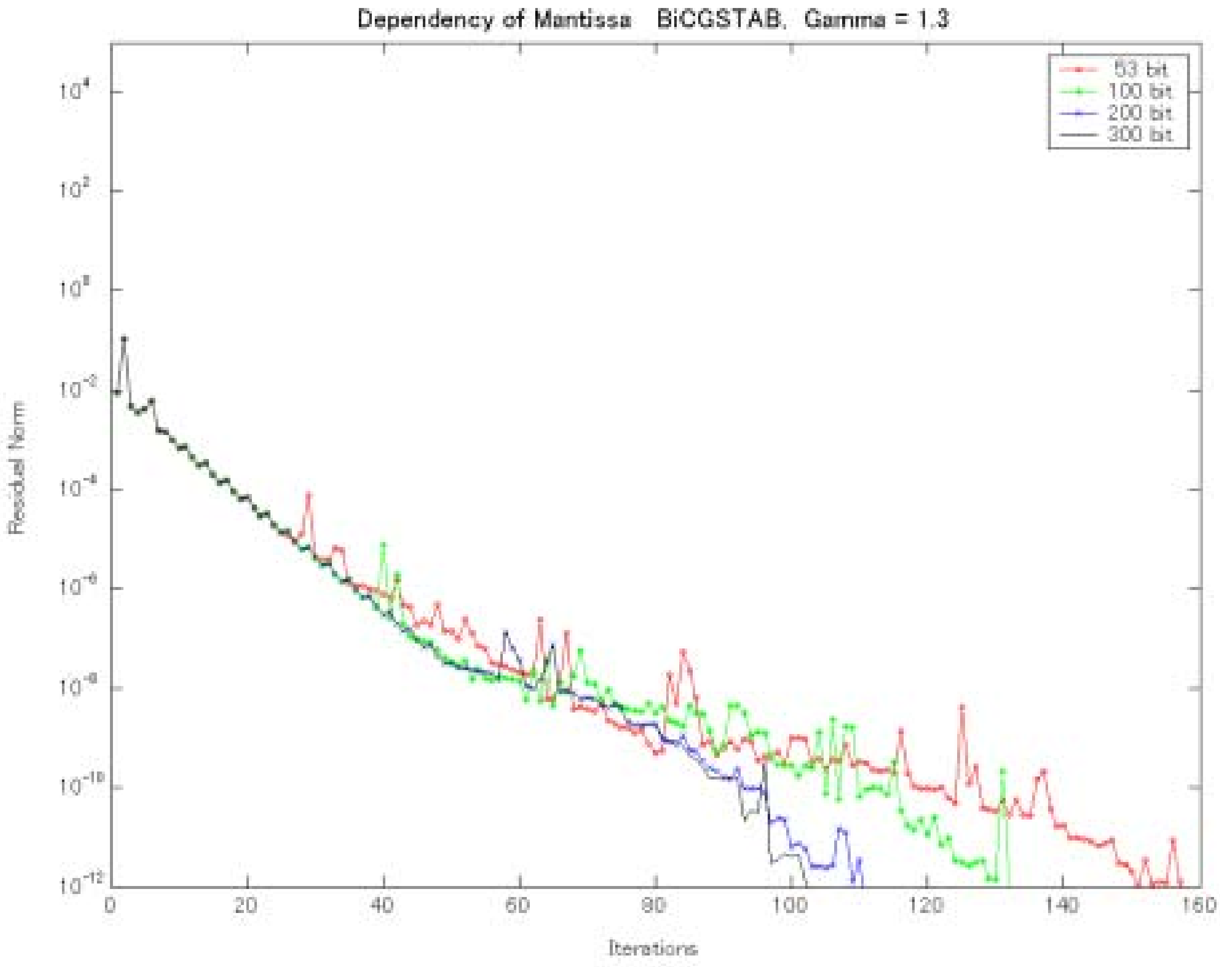


CGS Gamma = 1.3

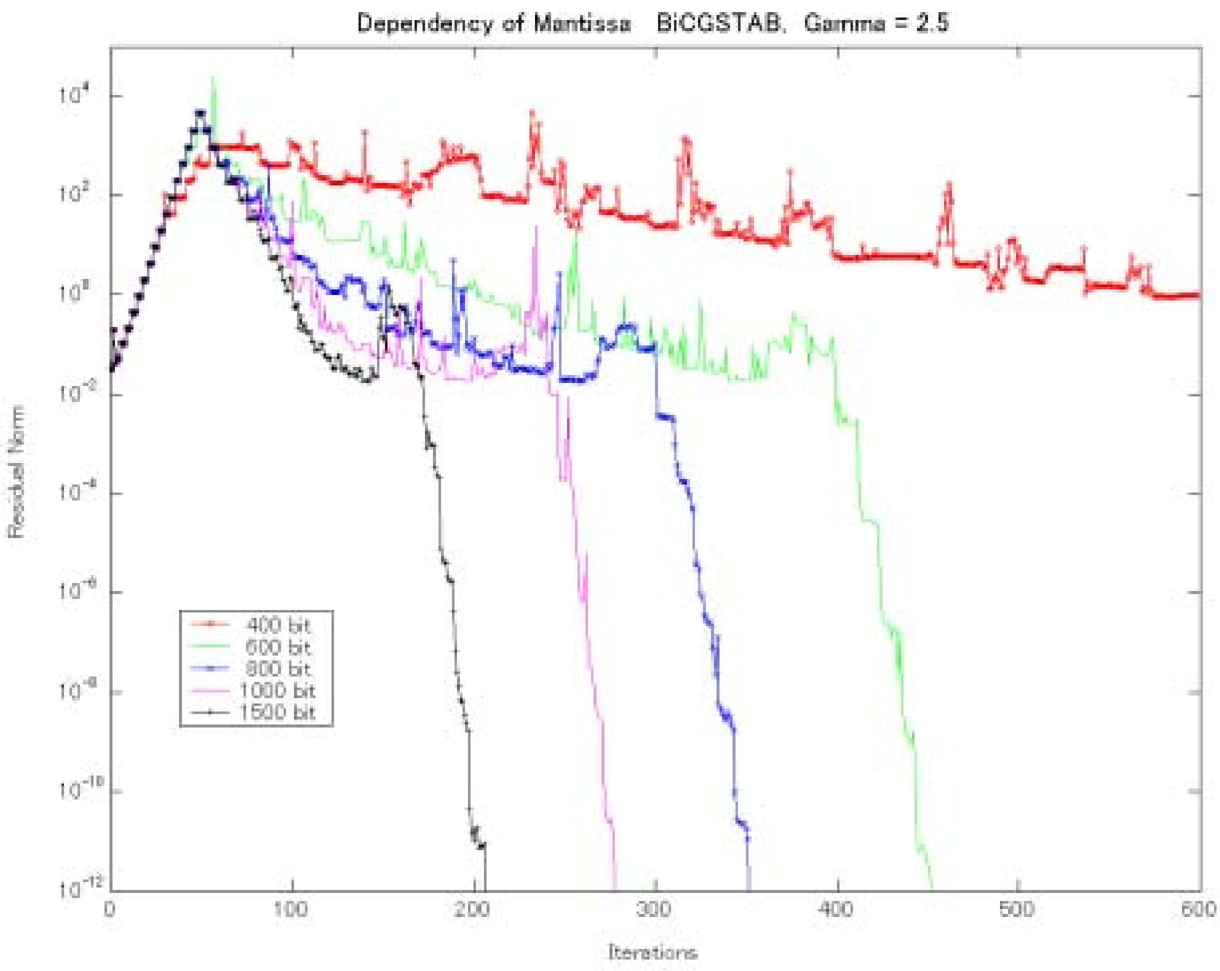


Dependency of Mantissa CGS, Gamma = 1.3
53 bit
100 bit
200 bit
300 bit
Residual Norm
Iterations
0
50
100
150
200
250
300

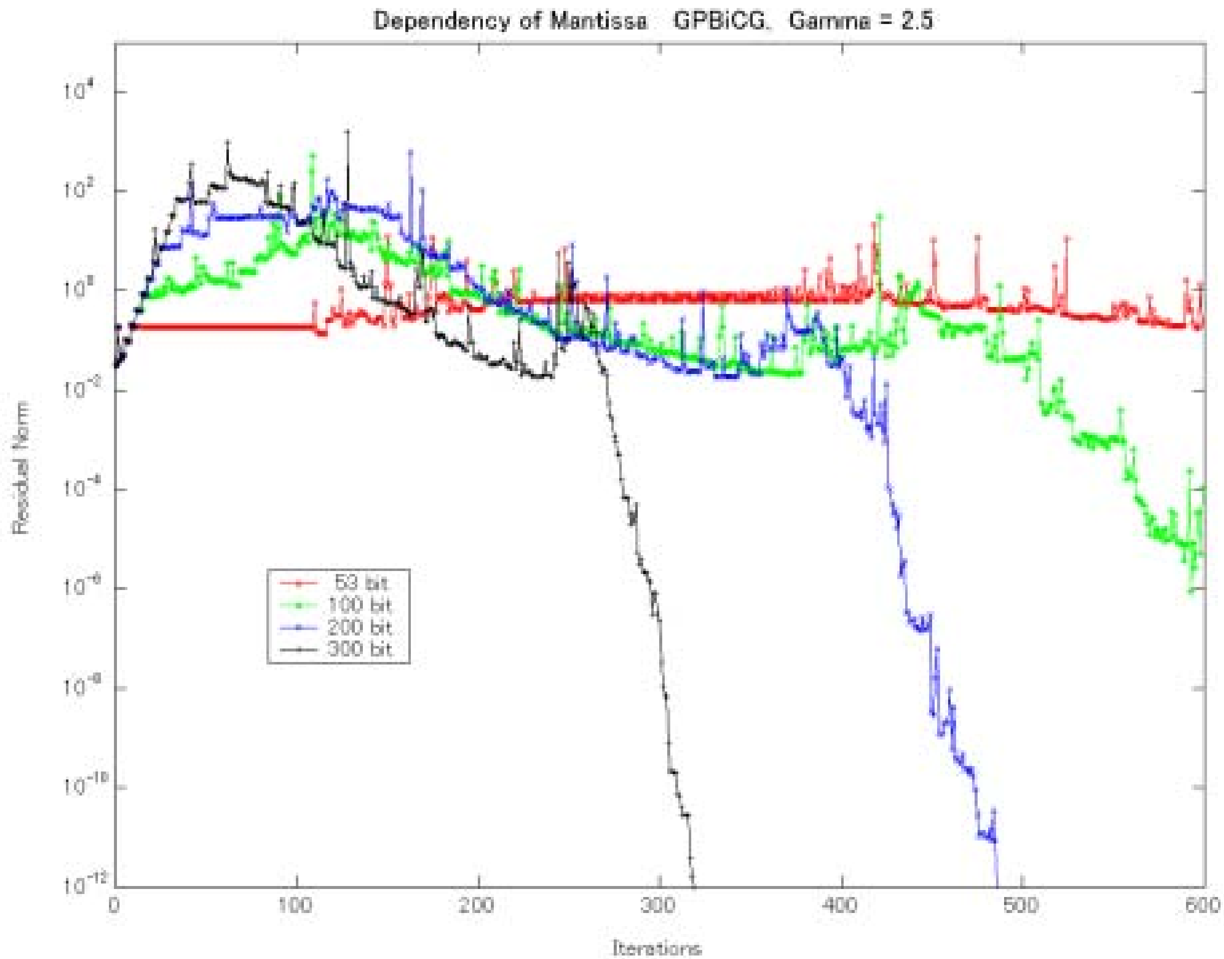
BiCGSTAB Gamma = 1.3



BiCGSTAB Gamma = 2.5



GPBiCG Gamma = 2.5



Observations

- Fast and smooth convergence are gained from More accurate computations.
- Required Mantissa is based on the problems:

BiCG	53 bit	for Gamma =	1.3
	100 bit	for	1.7
	200 bit	for	2.1
	200 bit	for	2.5

- Required Mantissa depends on Algorithms:

	BiCG	200 bit and 190 iterations	
	CGS	300 bit and 160	
x	BiCGSTAB	1500 bit and 210	
x	GPBiCG	300 bit and 310	(Gamma = 2.5)

Tools for Accurate Computing

- Multiple Precision Package (Gnu MP)
- Symbolic Computing (Computer Algebra)
- Interval Arithmetic
- Quadruple Floating-Point Operations

Sun Enterprise 3000 n=10000

	Double with ILU	Quad.	ratio
MATVEC	$4.69*10^{-3}$ (17.5%)	0.149 (24.7%)	31.7
MATVECT	$4.89*10^{-3}$ (18.2%)	0.159 (26.4%)	32.5
DAXPY DDOT DNORM2	$7.47*10^{-3}$ (27.8%)	0.290 (48.1%)	38.82
PSOLVE	$4.87*10^{-3}$ (18.1%)	—	—
PSOLVET	$4.92*10^{-3}$ (18.3%)	—	—
Total	$2.68*10^{-2}$	0.602	22.4

HITACHI MP5800 n=10000

	Double with ILU	Quad.	ratio
MATVEC	$0.135*10^{-2}$ (16.9%)	$0.468*10^{-2}$ (27.3%)	3.4
MATVECT	$0.134*10^{-2}$ (16.8%)	$0.472*10^{-2}$ (27.6%)	3.5
DAXPY DDOT DNORM2	$0.244*10^{-2}$ (30.6%)	$0.719*10^{-2}$ (42.0%)	2.9
PSOLVE	$0.146*10^{-2}$ (18.3%)	—	—
PSOLVET	$0.135*10^{-2}$ (16.9%)	—	—
Total	$0.796*10^{-2}$	$0.171*10^{-1}$	2.1

Fujitsu VPP800 n=10000

	Double with ILU	Quad.	ratio
MATVEC	$3.92*10^{-5}$ (1.06%)	$4.52*10^{-3}$ (11.9%)	115
MATVECT	$3.93*10^{-5}$ (1.07%)	$4.52*10^{-3}$ (11.9%)	115
DAXPY DDOT DNORM2	$1.21*10^{-3}$ (32.9%)	$2.86*10^{-2}$ (75.6%)	23.6
PSOLVE	$1.35*10^{-3}$ (36.7%)	—	—
PSOLVET	$1.03*10^{-3}$ (28.0%)	—	—
Total	$3.67*10^{-3}$	$3.78*10^{-2}$	10.2

HITACHI SR8000 n=10000

	Double with ILU	Quad.	ratio
MATVEC	$0.101*10^{-3}$ (3.5%)	$0.612*10^{-3}$ (11.3%)	6.0
MATVECT	$0.996*10^{-4}$ (3.4%)	$0.601*10^{-3}$ (11.1%)	6.0
DAXPY DDOT DNORM2	$0.781*10^{-3}$ (27.2%)	$0.419*10^{-2}$ (77.5%)	5.3
PSOLVE	$0.738*10^{-3}$ (25.7%)	—	—
PSOLVET	$0.115*10^{-2}$ (40.9%)	—	—
Total	$0.287*10^{-2}$	$0.540*10^{-2}$	1.8

Double with ILU vs Quadruple

	Double with ILU	Quad.	ratio
Sun (WS)	$0.268*10^{-1}$	0.602	22.4
VPP500 (Vector)	—	—	—
VPP300* (Vector)	$0.341*10^{-1}$	6.63	194
VPP800 (Vector)	$0.367*10^{-2}$	$0.378*10^{-1}$	10.2
SR8000 (SMP)	$0.287*10^{-2}$	$0.540*10^{-2}$	1.8
MP5800 (Mainframe)	$0.796*10^{-2}$	$0.171*10^{-1}$	2.1

Conclusion (tentative)

1. Fast and Smooth Convergence are gained from Accurate Computing.
2. Quadruple arithmetic is economically powerful tool, also easy and simple to use.
3. The best Algorithm varies depending on Computational Environment.
4. The simple Bi-CG is good for more Accurate Computing Environment.

Future works

1. Analysis: alpha, beta and other vectors.
2. Real problems:
3. Refinement of Implementation:
4. Find a good tool: easy and effective

Advertisement

- High performance should be used not only for “Speeding” but also “Quality of Computation”.
- To test effectiveness of Krylov Subspace Methods, try to change Computing Accuracy.
- Try to use Quadruple Arithmetic in High-Performance Computers and legacy machines.