kNaN

version 1.2b

Generated by Doxygen 1.5.8

Thu Apr 2 15:46:12 2009

Contents

Chapter 1

Main Page

1.1 Introduction

kNaN is a solver for systems of nonlinear equations based on Newton's Method with line search backtracking.

1.2 Features

Currently the solver can only work with explicit definitions of f() and jf() .

1.3 Future Work

Following methods are planned to be implemented:

- krylov
- matrix-free
- preconditioning
- freezed jacobian
- diffusion jacobian
- diffusion + freezed jacobian
- low order approximation to high order nonlinear system
- inexact newton

- multigrid
- numerical jacobian
- continuation
- fokkema vorst: store resulting F vectors
- singular systems

1.4 To do list for version 1.3:

- deltax'e gore de durma kriteri - s based
  1. absolute
  2. relative
- introduce analytical jacobian, numerical(one by one and column based)
- introduce jacobian vector multiplication - either directional derivative, full, or sparse storage schemes.
  1. always compare with the exact value for scaled, small and large for both F and X.
- rename mybeta as beta after numerical_libraries are excluded with the implementation of Krylov Solvers or LU decomposition routines

1.5 To do list for version 1.2:

- initial guess function (either parameter continuation or interpolation or just initialization)
- introduce analytical jacobian, numerical(one by one and column based)
- define weighted norm as a convergence option-option to compare different grids
- give a detailed analysis file with information on the iterations including the mesh size and weighted norm
- if relative residual is used then it is like weighted norms are used!
- reduce the number of extra F1 and F2 vectors. either by pointers or overwrites
- find a name for this version. this 1.2a - a letter might help :) !
- lambda plot!

1.6 Version History

1.6.1 version 1.2b - Multiplier

problem sets are added

code changes

basic inexact newton is implemented

separate matrix vector multiplier MatVecMult() is added

1.6.2 version 1.2a - Addition

bicgstab is added as a solver (matmul - no precon)

bicgstab_no_s is added as the default bicgstab solver (matmul- no precon)

1.6.3 version 1.1 - Expansion

absolute and relative tolerances are introduced. convergence options are introduced.

solverParameters and physicalParameters modules are added.

Debugging related extra information parameters are added.

module myFunctions is added. quadratic and cubic functions

are carried into this module.

machineps: machineps() - scalar function

calculates the machine eps

1.6.4 version 1.0 - Inception

In this version the most basic form of a line search backtracking algorithm is implemented. In this program the example problem 6.5.1 given in [1] is examined

1.6.5 References

[1] Numerical Methods for Unconstrained Optimization and Nonlinear Equations - J.E. Dennis, R.B. Schnabel - SIAM - 1996 corrected edition

[2] Iterative Methods for Linear and Nonlinear Equations - C.T. Kelley - SIAM - 1995
[3] Testing Unconstrained Optimization Software - J.J. More, B.S. Garbow, K.E. Hillstorm - TOMS 7 pp 136-140 - 1981

1.6.6 Resources

pdf version of the documentation is `here`.

code will be available here after version 1.5.

1.6.7 Developer

Erhan Turan

Chapter 2

Modules Index

2.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Module Documentation

4.1 localParameters Module Reference

Variables

- integer nx
- integer ny
- integer unk
- real(8) lambda
- real(8) Lx
- real(8) Ly

4.1.1 Variable Documentation

4.1.1.1 real(8) localParameters::lambda

Definition at line 7 of file localParameters.f90.

Referenced by f(), physicalParameters::initializePhysics(), and jf().

4.1.1.2 real(8) localParameters::Lx

Definition at line 9 of file localParameters.f90.

Referenced by f(), physicalParameters::initializePhysics(), jf(), and postProcessor::visualizer().

4.1.1.3 real(8) localParameters::Ly

Definition at line 9 of file localParameters.f90.

Referenced by f(), physicalParameters::initializePhysics(), jf(), and postProcessor::visualizer().

4.1.1.4 integer localParameters::nx

Definition at line 5 of file localParameters.f90.

Referenced by f(), physicalParameters::initializePhysics(), jf(), and postProcessor::visualizer().

4.1.1.5 integer localParameters::ny

Definition at line 5 of file localParameters.f90.

Referenced by f(), physicalParameters::initializePhysics(), jf(), and postProcessor::visualizer().

4.1.1.6 integer localParameters::unk

Definition at line 5 of file localParameters.f90.

Referenced by f(), physicalParameters::initializePhysics(), jf(), and postProcessor::visualizer().

4.2 main Module Reference

Variables

- integer i
- real(8), allocatable X

4.2.1 Variable Documentation

4.2.1.1 integer main::i

Definition at line 8 of file main.f90.

4.2.1.2 real(8),allocatable main::X

Definition at line 10 of file main.f90.

4.3 myFunctions Module Reference

Functions/Subroutines

- real(8) cubic (df0, f0, fprev1, fprev2, lambdaprev1, lambdaprev2)
- real(8) machineps ()
- real(8) quadratic (df0, f0, f1)

4.3.1 Function/Subroutine Documentation

4.3.1.1 real(8) myFunctions::cubic (real(8) *df0*, real(8) *f0*, real(8) *fprev1*, real(8) *fprev2*, real(8) *lambdaprev1*, real(8) *lambdaprev2*)

Definition at line 11 of file myFunctions.f90.

Referenced by newton().

Here is the caller graph for this function:

myFunctions::cubic
newton

4.3.1.2 real(8) myFunctions::machineps ()

Definition at line 33 of file myFunctions.f90.

4.3.1.3 real(8) myFunctions::quadratic (real(8) *df0*, real(8) *f0*, real(8) *f1*)

Definition at line 45 of file myFunctions.f90.

Referenced by newton().

Here is the caller graph for this function:

myFunctions::quadratic
newton

4.4 physicalParameters Module Reference

Functions/Subroutines

- subroutine initializePhysics ()

Variables

- integer n

4.4.1 Function/Subroutine Documentation

4.4.1.1 subroutine physicalParameters::initializePhysics ()

Definition at line 20 of file physicalParameters.f90.

References solverParameters::displaySolution, localParameters::lambda, solverParameters::linearTol, localParameters::Lx, localParameters::Ly, solverParameters::maxLinearIter, solverParameters::maxNewtonIter, n, solverParameters::NewtonConvergenceOption, solverParameters::nonlinearTolAbs, localParameters::nx, localParameters::ny, solverParameters::OFF, and localParameters::unk.

4.4.2 Variable Documentation

4.4.2.1 integer physicalParameters::n

Definition at line 8 of file physicalParameters.f90.

Referenced by initializePhysics().

4.5 postProcessor Module Reference

Functions/Subroutines

- subroutine visualizer (X, n)

4.5.1 Function/Subroutine Documentation

4.5.1.1 subroutine postProcessor::visualizer (real(8),dimension(n) *X*, integer *n*)

Definition at line 14 of file postProcessor.f90.

References localParameters::Lx, localParameters::Ly, localParameters::nx, localParameters::ny, and localParameters::unk.

4.6 solverParameters Module Reference

Variables

- logical, parameter TRUE = .true.
- logical, parameter FALSE = .false.
- logical, parameter ON = .true.
- logical, parameter OFF = .false.
- integer maxNewtonIter = 40

  maximum number of newton iterations

- integer maxBacktrackCount = 10

  maximum number of backtracks

- integer NewtonConvergenceOption = 1
- real(8) nonlinearTolAbs = 1e-6
- real(8) nonlinearTolRel = 1e-6
- logical newtonSuccess
- logical enableNewtonDebug = OFF
- logical enableNewtonDebugExtra = OFF
- logical enableNewtonBasic = ON
- logical NewtonConvergenceOptionArray

  —— burda kaldim

- integer maxLinearIter
- real(8) linearTol
- real(8) maxLinearTol = 1e-2
- logical displaySolution = ON
- logical printSolution = ON

4.6.1 Variable Documentation

4.6.1.1 logical solverParameters::displaySolution = ON

Definition at line 78 of file solverParameters.f90.

Referenced by physicalParameters::initializePhysics().

4.6.1.2 logical solverParameters::enableNewtonBasic = ON

Definition at line 51 of file solverParameters.f90.

Referenced by newton().

4.6.1.3 logical solverParameters::enableNewtonDebug = OFF

Definition at line 49 of file solverParameters.f90.

Referenced by newton().

4.6.1.4 logical solverParameters::enableNewtonDebugExtra = OFF

Definition at line 50 of file solverParameters.f90.

Referenced by newton().

4.6.1.5 logical,parameter solverParameters::FALSE = .false.

Definition at line 10 of file solverParameters.f90.

4.6.1.6 real(8) solverParameters::linearTol

Definition at line 70 of file solverParameters.f90.

Referenced by physicalParameters::initializePhysics().

4.6.1.7 integer solverParameters::maxBacktrackCount = 10

maximum number of backtracks

Definition at line 31 of file solverParameters.f90.

Referenced by newton().

4.6.1.8 integer solverParameters::maxLinearIter

Definition at line 68 of file solverParameters.f90.

Referenced by physicalParameters::initializePhysics(), and newton().

4.6.1.9 real(8) solverParameters::maxLinearTol = 1e-2

Definition at line 71 of file solverParameters.f90.

Referenced by newton().

4.6.1.10 integer solverParameters::maxNewtonIter = 40

maximum number of newton iterations

Definition at line 30 of file solverParameters.f90.

Referenced by physicalParameters::initializePhysics(), and newton().

4.6.1.11 integer solverParameters::NewtonConvergenceOption = 1

Definition at line 33 of file solverParameters.f90.

Referenced by physicalParameters::initializePhysics(), and newton().

4.6.1.12 logical solverParameters::NewtonConvergenceOptionArray

—— burda kaldim

Definition at line 53 of file solverParameters.f90.

4.6.1.13 logical solverParameters::newtonSuccess

Definition at line 48 of file solverParameters.f90.

Referenced by newton().

4.6.1.14 real(8) solverParameters::nonlinearTolAbs = 1e-6

Definition at line 38 of file solverParameters.f90.

Referenced by physicalParameters::initializePhysics(), and newton().

4.6.1.15 real(8) solverParameters::nonlinearTolRel = 1e-6

Definition at line 39 of file solverParameters.f90.

Referenced by newton().

4.6.1.16 logical,parameter solverParameters::OFF = .false.

Definition at line 11 of file solverParameters.f90.

Referenced by physicalParameters::initializePhysics().

4.6.1.17 logical,parameter solverParameters::ON = .true.

Definition at line 11 of file solverParameters.f90.

Referenced by newton().

4.6.1.18 logical solverParameters::printSolution = ON

Definition at line 79 of file solverParameters.f90.

4.6.1.19 logical,parameter solverParameters::TRUE = .true.

Definition at line 10 of file solverParameters.f90.

Chapter 5

File Documentation

5.1 E:/turkuaz/krylov/codes/newton/1.2b/linear/krylov/bicgstab.f90 File Reference

Functions/Subroutines

- subroutine bicgstab (n, A, b, x, tol, maxIter, k)

5.1.1 Function Documentation

5.1.1.1 subroutine bicgstab (integer *n*, real(8),dimension(n,n) *A*, real(8),dimension(n) *b*, real(8),dimension(n) *x*, real(8) *tol*, integer *maxIter*, integer *k*)

Definition at line 1 of file bicgstab.f90.

5.2 E:/turkuaz/krylov/codes/newton/1.2b/linear/krylov/bicgstab_no_s.f90 File Reference

Functions/Subroutines

- subroutine bicgstab_no_s (A, b, x, n, tol, mainloop, k)

5.2.1 Function Documentation

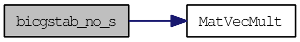
5.2.1.1 subroutine bicgstab_no_s (real(8),dimension(n,n) *A*, real(8),dimension(n) *b*, real(8),dimension(n) *x*, integer *n*, real(8) *tol*, integer *mainloop*, integer *k*)

Definition at line 1 of file bicgstab_no_s.f90.

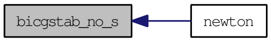
References MatVecMult().

Referenced by newton().

Here is the call graph for this function:



Here is the caller graph for this function:



5.3 E:/turkuaz/krylov/codes/newton/1.2b/main.f90 File Reference

Modules

- module main

Variables

- integer main::i
- real(8), allocatable main::X

5.4 E:/turkuaz/krylov/codes/newton/1.2b/newton/modules/myFunctions.f9 File Reference

Modules

- module myFunctions

Functions/Subroutines

- real(8) myFunctions::cubic (df0, f0, fprev1, fprev2, lambdaprev1, lambdaprev2)
- real(8) myFunctions::machineps ()
- real(8) myFunctions::quadratic (df0, f0, f1)

5.5 E:/turkuaz/krylov/codes/newton/1.2b/newton/modules/solverParameters.f90 File Reference

Modules

- module solverParameters

Variables

- logical, parameter solverParameters::TRUE = .true.
- logical, parameter solverParameters::FALSE = .false.
- logical, parameter solverParameters::ON = .true.
- logical, parameter solverParameters::OFF = .false.
- integer solverParameters::maxNewtonIter = 40

  maximum number of newton iterations

- integer solverParameters::maxBacktrackCount = 10

  maximum number of backtracks

- integer solverParameters::NewtonConvergenceOption = 1
- real(8) solverParameters::nonlinearTolAbs = 1e-6
- real(8) solverParameters::nonlinearTolRel = 1e-6
- logical solverParameters::newtonSuccess
- logical solverParameters::enableNewtonDebug = OFF
- logical solverParameters::enableNewtonDebugExtra = OFF
- logical solverParameters::enableNewtonBasic = ON
- logical solverParameters::NewtonConvergenceOptionArray

  —— burda kaldim

- integer solverParameters::maxLinearIter
- real(8) solverParameters::linearTol
- real(8) solverParameters::maxLinearTol = 1e-2
- logical solverParameters::displaySolution = ON
- logical solverParameters::printSolution = ON

5.6 E:/turkuaz/krylov/codes/newton/1.2b/newton/newton.f90 File Reference

Functions/Subroutines

- subroutine newton (X, n)

  Newton-Raphson Method.

5.6.1 Function Documentation

5.6.1.1 subroutine newton (real(8),dimension(n) *X*, integer *n*)

Newton-Raphson Method.

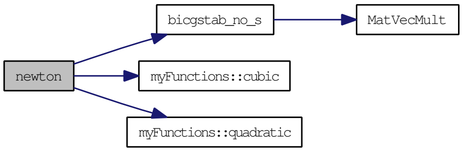
Parameters:

n size of the unknown vector

Definition at line 7 of file newton.f90.

References bicgstab_no_s(), myFunctions::cubic(), solverParameters::enableNewtonBasic, solverParameters::enableNewtonDebug, solverParameters::enableNewtonDebugExtra, solverParameters::maxBacktrackCount, solverParameters::maxLinearIter, solverParameters::maxLinearTol, solverParameters::maxNewtonIter, solverParameters::NewtonConvergenceOption, solverParameters::newtonSuccess, solverParameters::nonlinearTolAbs, solverParameters::nonlinearTolRel, solverParameters::ON, and myFunctions::quadratic().

Here is the call graph for this function:



5.7 E:/turkuaz/krylov/codes/newton/1.2b/physics/bratu/anaJ/f.f90 File Reference

Functions/Subroutines

- subroutine f (X, func, n)

5.7.1 Function Documentation

5.7.1.1 subroutine f (real(8),dimension(n) *X*, real(8),dimension(n) *func*, integer *n*)

Definition at line 2 of file f.f90.

References localParameters::lambda, localParameters::Lx, localParameters::Ly, localParameters::nx, localParameters::ny, and localParameters::unk.

5.8 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/1/f.f90 File Reference

Functions/Subroutines

- subroutine f (X, func, n)

5.8.1 Function Documentation

5.8.1.1 subroutine f (real(8),dimension(n) *X*, real(8),dimension(n) *func*, integer *n*)

Definition at line 2 of file f.f90.

5.9 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/2/f.f90 File Reference

Functions/Subroutines

- subroutine f (X, func, n)

5.9.1 Function Documentation

5.9.1.1 subroutine f (real(8),dimension(n) *X*, real(8),dimension(n) *func*, integer *n*)

Definition at line 3 of file f.f90.

5.10 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/3/f.f90 File Reference

Functions/Subroutines

- subroutine f (X, func, n)

5.10.1 Function Documentation

5.10.1.1 subroutine f (real(8),dimension(n) *X*, real(8),dimension(n) *func*, integer *n*)

Definition at line 3 of file f.f90.

5.11 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/4/f.f90 File Reference

Functions/Subroutines

- subroutine f (X, func, n)

5.11.1 Function Documentation

5.11.1.1 subroutine f (real(8),dimension(n) *X*, real(8),dimension(n) *func*, integer *n*)

Definition at line 3 of file f.f90.

5.12 E:/turkuaz/krylov/codes/newton/1.2b/physics/bratu/anaJ/initialGuess File Reference

Functions/Subroutines

- subroutine initialGuess (X, n)

5.12.1 Function Documentation

5.12.1.1 subroutine initialGuess (real(8),dimension(n) *X*, integer *n*)

Definition at line 1 of file initialGuess.f90.

5.13 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/1/initialGuess.f90 File Reference

Functions/Subroutines

- subroutine initialGuess (X, n)

5.13.1 Function Documentation

5.13.1.1 subroutine initialGuess (real(8),dimension(n) *X*, integer *n*)

Definition at line 1 of file initialGuess.f90.

5.14 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/2/initialGuess.f90 File Reference

Functions/Subroutines

- subroutine initialGuess (X, n)

5.14.1 Function Documentation

5.14.1.1 subroutine initialGuess (real(8),dimension(n) *X*, integer *n*)

Definition at line 1 of file initialGuess.f90.

5.15 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/3/initialGuess.f90 File Reference

Functions/Subroutines

- subroutine initialGuess (X, n)

5.15.1 Function Documentation

5.15.1.1 subroutine initialGuess (real(8),dimension(n) *X*, integer *n*)

Definition at line 1 of file initialGuess.f90.

5.16 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/4/initialGuess.f90 File Reference

Functions/Subroutines

- subroutine initialGuess (X, n)

5.16.1 Function Documentation

5.16.1.1 subroutine initialGuess (real(8),dimension(n) *X*, integer *n*)

Definition at line 1 of file initialGuess.f90.

5.17 E:/turkuaz/krylov/codes/newton/1.2b/physics/bratu/anaJ/jf.f90 File Reference

Functions/Subroutines

- subroutine jf (X, JAC, n)

5.17.1 Function Documentation

5.17.1.1 subroutine jf (real(8),dimension(n) *X*, real(8),dimension(n,n) *JAC*, integer *n*)

Definition at line 2 of file jf.f90.

References localParameters::lambda, localParameters::Lx, localParameters::Ly, localParameters::nx, localParameters::ny, and localParameters::unk.

5.18 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/1/jf.f90 File Reference

Functions/Subroutines

- subroutine jf (X, JAC, n)

5.18.1 Function Documentation

5.18.1.1 subroutine jf (real(8),dimension(n) *X*, real(8),dimension(n,n) *JAC*, integer *n*)

Definition at line 2 of file jf.f90.

5.19 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/2/jf.f90 File Reference

Functions/Subroutines

- subroutine jf (X, JAC, n)

5.19.1 Function Documentation

5.19.1.1 subroutine jf (real(8),dimension(n) *X*, real(8),dimension(n,n) *JAC*, integer *n*)

Definition at line 3 of file jf.f90.

5.20 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/3/jf.f90 File Reference

Functions/Subroutines

- subroutine jf (X, JAC, n)

5.20.1 Function Documentation

5.20.1.1 subroutine jf (real(8),dimension(n) *X*, real(8),dimension(n,n) *JAC*, integer *n*)

Definition at line 3 of file jf.f90.

5.21 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/4/jf.f90 File Reference

Functions/Subroutines

- subroutine jf (X, JAC, n)

5.21.1 Function Documentation

5.21.1.1 subroutine jf (real(8),dimension(n) *X*, real(8),dimension(n,n) *JAC*, integer *n*)

Definition at line 3 of file jf.f90.

5.22 E:/turkuaz/krylov/codes/newton/1.2b/physics/bratu/anaJ/localParame File Reference

Modules

- module localParameters

Variables

- integer localParameters::nx
- integer localParameters::ny
- integer localParameters::unk
- real(8) localParameters::lambda
- real(8) localParameters::Lx
- real(8) localParameters::Ly

5.23 E:/turkuaz/krylov/codes/newton/1.2b/physics/bratu/anaJ/physicalParameters. File Reference

Modules

- module physicalParameters

Functions/Subroutines

- subroutine physicalParameters::initializePhysics ()

Variables

- integer physicalParameters::n

5.24 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/1/physicalParamete File Reference

Modules

- module physicalParameters

Functions/Subroutines

- subroutine physicalParameters::initializePhysics ()

5.25 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/2/physicalParameters.f90 File Reference

Modules

- module physicalParameters

Functions/Subroutines

- subroutine physicalParameters::initializePhysics ()

5.26 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/3/physicalParamete File Reference

Modules

- module physicalParameters

Functions/Subroutines

- subroutine physicalParameters::initializePhysics ()

5.27 E:/turkuaz/krylov/codes/newton/1.2b/physics/test/4/physicalParameters.f90 File Reference

Modules

- module physicalParameters

Functions/Subroutines

- subroutine physicalParameters::initializePhysics ()

5.28 E:/turkuaz/krylov/codes/newton/1.2b/physics/bratu/anaJ/postProcess File Reference

Modules

- module postProcessor

Functions/Subroutines

- subroutine postProcessor::visualizer (X, n)

5.29 E:/turkuaz/krylov/codes/newton/1.2b/readme.txt File Reference

5.30 E:/turkuaz/krylov/codes/newton/1.2b/routines/MatVecMult.f90 File Reference

Functions/Subroutines

- subroutine MatVecMult (A, v, w, n)

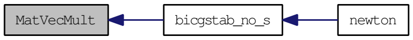
5.30.1 Function Documentation

5.30.1.1 subroutine MatVecMult (real(8),dimension(n,n) *A*, real(8),dimension(n) *v*, real(8),dimension(n) *w*, integer *n*)

Definition at line 1 of file MatVecMult.f90.

Referenced by bicgstab_no_s().

Here is the caller graph for this function:



Index