Math128B: Numerical Analysis

Programming Project, Due April 30, 2018

Choose any one of the following 4 projects, details of which are in Week 11 Lecture Notes. Turn in a .m file Projectxyyyy.m, where `x` is the project number and `yyyy` is your student ID.

Email your .m file to your GSI by 11:59PM, April 30, 2018.

Proj. 1 Solving linear equations with Gaussian Elimination

- Implement GE, GEPP, GECP, with and without iterative refinement.
- Compare with `matlab` \ function in terms of accuracy (residual norm) and execution time.
- Input/Output format

```
function [GEPP,GECP] = Project1yyyy(A,b,tau)
```

where GEPP is an object with the following fields:

GEPP.P	Row permutation
GEPP.L	L matrix
GEPP.U	U matrix
GEPP.x	solution
GEPP.xref	refined solution
GEPP.rnorm	2-norm of residual
GEPP.time	run time in seconds
GEPP.matlab.rnorm	2-norm of residual in `matlab` solution
GECP.matlab.time	run time in seconds of `matlab` solution

GECP is an object with the following fields:

GECP.Pr	Row permutation
GECP.Pc	Column permutation
GECP.L	L matrix
GECP.U	U matrix
GECP.x	solution
GECP.xref	refined solution
GECP.rnorm	2-norm of residual
GECP.time	run time in seconds
GECP.matlab.rnorm	2-norm of residual in `matlab` solution
GECP.matlab.time	run time in seconds of `matlab` solution

Proj. 2 QR Algorithm for Symmetric Eigenproblem

- Householder reduction to tridiagonal
- Implicit bulge Chasing scheme for tridiagonal QR Algorithm.
- Recover eigenvectors.
- Input/Output format

```
function EIG = Project2yyyy(A,tau)
```

where EIG is an object with the following fields:

EIG.H	Product of all Householder reflections
EIG.T	tridiagonal matrix
EIG.QT	Eigenvector matrix of T
EIG.Q	Eigenvector matrix of A
EIG.D	Vector of all eigenvalues of A
EIG.res_norm	2-norm of $A \times$ (EIG.Q) $-$ (EIG.Q) $\times$ **diag** (EIG.D)
EIG.orth_norm	2-norm of $(\text{EIG.Q})^T \times$ (EIG.Q) $- I$
EIG.time	run time in seconds
EIG.matlab.time	run time in seconds of `matlab` function `eig`

Proj. 3 QR Algorithm for SVD

- Householder reduction to bidiagonal
- Implicit bulge Chasing scheme for bidiagonal QR Algorithm.
- Recover singular vectors.
- Input/Output format

```
function SVD = Project3yyyy(A,tau)
```

where SVD is an object with the following fields:

SVD.G	Product of all left Householder reflections
SVD.H	Product of all right Householder reflections
SVD.B	bidiagonal matrix
SVD.U	Left singular vector matrix of A
SVD.V	Right singular vector matrix of A
SVD.S	Vector of all singular values of A
SVD.res_norm	2-norm of $A \times$ (SVD.V) $-$ (SVD.U) $\times$ **diag** (SVD.S)
SVD.U_orth_norm	2-norm of $(\text{SVD.U})^T \times$ (SVD.U) $- I$
SVD.V_orth_norm	2-norm of $(\text{SVD.V})^T \times$ (SVD.V) $- I$
SVD.time	run time in seconds
SVD.matlab.time	run time in seconds of `matlab` function `svd`

Proj. 4 Quasi-Newton Methods (BFGS) for $\mathbf{min}_{\mathbf{x},c}\, g(\mathbf{x}, \mathbf{A}, \mathbf{y}, c, \lambda)$

- Implement BFGS and L-BFGS
- Run BFGS and L-BFGS with logistic objective function

$$g(\mathbf{x}, \mathbf{A}, \mathbf{y}, c, \lambda) = -\frac{1}{n}\sum_{i=1}^{n} \mathbf{log}\left(\phi\left(y_i\left(\mathbf{a}_i^T\mathbf{x} + c\right)\right)\right) \; + \; \frac{\lambda}{2}\|\mathbf{x}\|_2^2,$$

$$\text{where } \mathbf{x} \in \mathcal{R}^p, \quad c \in \mathcal{R}, \quad \mathbf{A} = \begin{pmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_n^T \end{pmatrix} \in \mathcal{R}^{n\times p}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathcal{R}^n,$$

and $\phi(t) = (1 + \mathbf{exp}(-t))^{-1}$. For numerical stability, however, the function $\mathbf{log}(\phi(\cdot))$ should be computed as follows:

$$\mathbf{log}(\phi(t)) = \mathbf{min}(0, t) - \mathbf{log}(1 + \mathbf{exp}(-|t|)).$$

Checkout more details about logistic function at
`http://fa.bianp.net/blog/2013/numerical-optimizers-for-logistic-regression/`

- Input/Output format

```
function [BFGS,LBFGS] = Project4yyyy(A,y, lambda, m, IT)
```

where `IT` is the number of BFGS and L-BFGS iterations. BFGS is an object with the following fields:

BFGS.flg	0 if BFGS succeeds, and 1 otherwise
BFGS.x	vector x computed by BFGS
BFGS.c	scalar computed by BFGS
BFGS.obj	computed optimal objective value.

LBFGS is an object with the following fields:

LBFGS.flg	0 if LBFGS succeeds, and 1 otherwise
LBFGS.x	vector x computed by LBFGS
LBFGS.c	scalar computed by LBFGS
LBFGS.obj	computed optimal objective value.

You can set $\mathcal{B}_0^{-1}$ to be a small multiple of the identity matrix.