Octave Programming Tutorial/Linear algebra

Functions

- `det(A)` computes the determinant of the matrix *A*.
- `lambda = eig(A)` returns the eigenvalues of `A` in the vector `lambda`, and
- `[V, lambda] = eig(A)` also returns the eigenvectors in `V` but `lambda` is now a matrix whose diagonals contain the eigenvalues. This relationship holds true (within round off errors) `A = V*lambda*inv(V)`.
- `inv(A)` computes the inverse of non-singular matrix *A*. Note that calculating the inverse is often 'not' necessary. See the next two operators as examples. Note that in theory `A*inv(A)` should return the identity matrix, but in practice, there may be some round off errors so the result may not be exact.
- `A / B` computes *X* such that $XB = A$. This is called right division and is done without forming the inverse of *B*.
- `A \ B` computes *X* such that $AX = B$. This is called left division and is done without forming the inverse of *A*.
- `norm(A, p)` computes the *p*-norm of the matrix (or vector) *A*. The second argument is optional with default value $p = 2$.
- `rank(A)` computes the (numerical) rank of a matrix.
- `trace(A)` computes the trace (sum of the diagonal elements) of *A*.
- `expm(A)` computes the matrix exponential of a square matrix. This is defined as
$$I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots$$
- `logm(A)` computes the matrix logarithm of a square matrix.
- `sqrtm(A)` computes the matrix square root of a square matrix.

Below are some more linear algebra functions. Use `help` to find out more about them.

- `balance` (eigenvalue balancing),
- `cond` (condition number),
- `dmult` (computes diag(x) * A efficiently),
- `dot` (dot product),
- `givens` (Givens rotation),
- `kron` (Kronecker product),
- `null` (orthonormal basis of the null space),
- `orth` (orthonormal basis of the range space),
- `pinv` (pseudoinverse),
- `syl` (solves the Sylvester equation).

Factorizations

- `R = chol(A)` computes the Cholesky factorization of the symmetric positive definite matrix *A*, i.e. the upper triangular matrix *R* such that $R^T R = A$.
- `[L, U] = lu(A)` computes the LU decomposition of *A*, i.e. *L* is lower triangular, *U* upper triangular and $A = LU$.
- `[Q, R] = qr(A)` computes the QR decomposition of *A*, i.e. *Q* is orthogonal, *R* is upper triangular and $A = QR$.

Below are some more available factorizations. Use `help` to find out more about them.

- `qz` (generalized eigenvalue problem: QZ decomposition),
- `qzhess` (Hessenberg-triangular decomposition),
- `schur` (Schur decomposition),
- `svd` (singular value decomposition),
- `housh` (Householder reflections),
- `krylov` (Orthogonal basis of block Krylov subspace).

Return to the Octave Programming Tutorial index
