Journal of Information Processing Vol.23 No.2 171–175 (Mar. 2015)

[DOI: 10.2197/ipsjjip.23.171]

Regular Paper

A Fast Computation of 3 × 3 Matrix Exponentials and its Application in CG

Kohei Matsushita[1,a)] Hiroyasu Hamada[2,b)]

Received: June 30, 2014, Accepted: December 3, 2014

Abstract: It is often useful to compute a lot of matrix exponentials in computer graphics (CG). The exponential of a matrix is used for the smooth deformation of 2D or 3D meshed CG objects. Hence, we need to compute a large number of the exponentials of 3×3 rotational matrices and 3×3 real symmetric matrices. For rotational matrices, Rodrigues' formula is known to compute their exponentials. We investigated the polynomial methods introduced by Moler and Van Loan to compute an exponential of 3×3 real symmetric matrices, and we introduce an algorithm for eigenvalues of 3×3 real symmetric matrices. We introduce a simple formula for the matrix exponential of a 3×3 real symmetric matrix using a formula introduced by Kaji et al. in 2013 and Viète's Formula. Since our matrix exponential algorithm do not use eigenvectors, we are able to reduce the computational cost using a fast eigenvalue computation algorithm. Then, we incorporated our implementation into a shape deforming tool developed by Kaji et al. As a result, we achieved a notable performance improvement. In fact we show our algorithms for matrix exponentials is about 76% faster than a standard algorithm for given 3×3 real symmetric matrices. For the deformation of a CG model, our algorithm was about 19% faster than a standard algorithm.

Keywords: numerical computation and analysis, linear algebra and matrix computations

1. Introduction

Matrix exponential computations have been used for many applications in computer graphics (CG). For example, *mesh-based inverse kinematics* provides a tool that simplifies posing task [10]. In this algorithm, a matrix exponential is used to calculate a deformation gradient of triangular meshes. Alexa investigated a new interpolation between transformations using operations, addition and scalar multiplication, which create weighted combination of transformations and interpolation [1]. For 2D shape interpolation and deformation, there are some algorithms that preserve rigidity [2], [5]. Kaji et al. improved those algorithms by using matrix exponentials [7]. Therefore, matrix exponentials are often used for smooth deformations of 2D or 3D meshed CG objects.

Our goal is to provide a fast computation of matrix exponentials. Our motivation is to improve the performance of many applications using exponentials of 3×3 rotational matrices and 3×3 real symmetric matrices introduced in Refs. [6], [7], [10]. For the rotational matrices, Rodrigues' formula [3] is known to compute their exponentials. In this paper, we consider an improvement of an exponential of a 3×3 real symmetric matrix rather than a rotation matrix.

In general, there are many approaches for computing $n \times n$ matrix exponentials. Moler and Van Loan provided several methods to compute the matrix exponentials [9]. We are interested in the fast computation of the matrix exponential of a special case such as 3×3 real matrices used for the affine transformations [6], [8]. In this paper, We investigate the spectral decomposition method in Ref. [9] focusing on 3×3 real symmetric matrices. Our algorithm needs only eigenvalues of a given matrix. We note a method using diagonalization have to compute eigenvectors in addition to eigenvalues. Hence, our algorithm can compute matrix exponentials efficiently. In addition, we consider a faster algorithm for computing the eigenvalues of a 3×3 real symmetric matrix. For a given matrix, we need to solve the characteristic equation of the matrix to calculate its eigenvalues. We use Viète's formula to compute the eigenvalues of a 3×3 real symmetric matrix. As a result, we introduce simple formulas of eigenvalues just using the trace and determinant of a given matrix. To evaluate the performance of our algorithm, we compare the average runtimes for matrix exponentials and eigenvalues.

2. Algorithm for Matrix Exponential

Let $M_n(\mathbb{R})$ be the set of $n \times n$ matrices.

Definition 2.1. *The set of* 3×3 *real symmetric matrices is defined by* $\mathrm{Sym}(3) := \{S \mid S = {}^tS \in M_3(\mathbb{R})\}$*, where* ${}^tS$ *is the transpose of* S.

Definition 2.2. *For* $S \in \mathrm{Sym}(3)$*, the matrix exponential of* S *is defined by*

$$\exp(S) = \sum_{k=0}^{\infty} \frac{S^k}{k!}.$$

Although the sum of the infinite series converges, their rate of convergence may not be so high. So we can not have an efficient algorithm by direct computations.

[1] Kyushu University, Fukuoka 819–0395, Japan
[2] National Institute of Technology, Sasebo College, Sasebo, Nagasaki 857–1193, Japan
[a)] k-matsushita@math.kyushu-u.ac.jp
[b)] h-hamada@sasebo.ac.jp

2.1 Algorithm1. Diagonalization

First, we introduce the method using diagonalization. This method helps to compute their exponential easily.

Proposition 2.1. *Let S be a real symmetric matrix. Then, S is decomposed by an orthogonal matrix P and a diagonal matrix D, and S is described as $S = PD{}^tP$.*

Proposition 2.2. *Let S be an element of* Sym(3)*. Then, the matrix exponential of S is $\exp(S) = P\exp(D){}^tP$, where P is an orthogonal matrix and D is a diagonal matrix. P and D satisfy $S = PD{}^tP$.*

Proof. Since S is a real symmetric matrix, S is also diagonalized with an orthogonal matrix P and a diagonal matrix D such that $S = PD{}^tP$.

From Definition 2.2, $\exp(S)$ is

$$\exp(S) = \exp(PD{}^tP) = \sum_{k=0}^{\infty} \frac{(PD{}^tP)^k}{k!}$$
$$= P\left(\sum_{k=0}^{\infty} \frac{D^k}{k!}\right){}^tP = P\exp(D){}^tP.$$

□

So we can make an algorithm of a matrix exponential using the diagonalization. The algorithm needs to compute eigenvalues (D) and eigenvectors (P) to compute the matrix exponential. The computational cost of eigenvectors however is more expensive than the cost of eigenvalues. Next, we review the faster algorithm in Ref. [6] to improve this drawback. This method does not need to compute eigenvectors and is faster than the method using diagonalization.

2.2 Algorithm2. Spectral Decomposition

We describe the algorithm using spectral decomposition [6], [8]. Let λ_1, λ_2 and λ_3 be the eigenvalues of $S \in \mathrm{Sym}(3)$. They are the roots of the characteristic polynomial of S. The characteristic polynomial of S is defined as follows.

Definition 2.3. *Let $\phi_S(\lambda)$ be the characteristic polynomial of $S \in$ Sym(3). $\phi_S(\lambda)$ is defined as $\phi_S(\lambda) = \det(\lambda E - S)$, where* det *is the determinant operation.*

From the Cayley-Hamilton theorem, the following proposition is satisfied.

Proposition 2.3. *Let $\phi_S(\lambda)$ be the characteristic polynomial of $S \in$ Sym(3) and we substitute S for λ and the identity matrix for 1 in this polynomial. Then, $\phi_S(S)$ is equal to 0.*

Since $\phi_S(\lambda)$ is a third degree polynomial in λ, $\phi_S(S)$ is also a third degree polynomial in S. Therefore $\frac{S^k}{k!}$ can be described as $\frac{S^k}{k!} = Q_k\phi_S(S) + R_k$, where Q_k is a polynomial in S, and R_k is an at most second degree polynomial in S. Hence, $\exp(S) = \sum_{k=0}^{\infty}(Q_k\phi_S(S) + R_k)$. Then, $\sum_{k=0}^{\infty} Q_k\phi_S(S)$ is equal to 0 from the Cayley-Hamilton theorem, and $\sum_{k=0}^{\infty} R_k$ is an at most second degree polynomial in S. As a result, $\exp(S)$ can be described as $\exp(S) = xS^2 + yS + zE$, where E is the 3×3 identity matrix, and x, y and z are elements of $\mathbb{R}$.

Hence, $\exp(S)$ can be represented by a second degree polynomial in S. Next step is to decide $x, y, z \in \mathbb{R}$ to compute $\exp(S)$. In Proposition 2.2, $\exp(S)$ can be described as $\exp(S) = P\exp(D){}^tP$.

On the other hand, $xS^2+yS+zE$ is described as $xS^2+yS+zE = P(xD^2 + yD + zE){}^tP$. Therefore, $\exp(D) = xD^2 + yD + zE$.

Let λ_1, λ_2 and λ_3 be the eigenvalues of S. The elements of D are the eigenvalues of S. Then, we have

$$\exp(D) = \begin{pmatrix} e^{\lambda_1} & 0 & 0 \\ 0 & e^{\lambda_2} & 0 \\ 0 & 0 & e^{\lambda_3} \end{pmatrix},$$

$$xD^2 + yD + zE = \begin{pmatrix} x\lambda_1^2 + y\lambda_1 + z & 0 & 0 \\ 0 & x\lambda_2^2 + y\lambda_2 + z & 0 \\ 0 & 0 & x\lambda_3^2 + y\lambda_3 + z \end{pmatrix}.$$

Hence, we can decide x, y and z to solve the following equation.

$$\begin{pmatrix} e^{\lambda_1} \\ e^{\lambda_2} \\ e^{\lambda_3} \end{pmatrix} = \begin{pmatrix} \lambda_1^2 & \lambda_1 & 1 \\ \lambda_2^2 & \lambda_2 & 1 \\ \lambda_3^2 & \lambda_3 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

The values of x, y and z depend on multiplicity of the eigenvalues of S. We consider two eigenvalues λ and λ' are same if $|\lambda - \lambda'| < 10^{-6}$.

Case 1. When all the three eigenvalues are same, put

$$x = y = 0,\ z = \exp(\lambda_1).$$

Case 2. When two of them are same, that is $\lambda_1 = \lambda_2$, put

$$x = 0,\ y = s - t,\ z = t\lambda_2 - s\lambda_3,$$

where

$$s = \frac{\exp(\lambda_2)}{\lambda_2 - \lambda_3},\ t = \frac{\exp(\lambda_3)}{\lambda_2 - \lambda_3}.$$

Case 3. When all of them are distinct, put

$$x = s + t + u,$$
$$y = -s(\lambda_2 + \lambda_3) + t(\lambda_3 + \lambda_1) + u(\lambda_1 + \lambda_2),$$
$$z = s\lambda_2\lambda_3 - t\lambda_3\lambda_1 - u\lambda_1\lambda_2,$$

where

$$s = \frac{\exp(\lambda_1)}{(\lambda_1 - \lambda_2)(\lambda_1 - \lambda_3)},$$
$$t = \frac{\exp(\lambda_2)}{(\lambda_2 - \lambda_3)(\lambda_1 - \lambda_2)},$$
$$u = \frac{\exp(\lambda_3)}{(\lambda_2 - \lambda_3)(\lambda_3 - \lambda_1)}.$$

Therefore, we have a simple formula for x, y and z. Our algorithm needs to compute only eigenvalues of a 3×3 real symmetric matrix so that we can compute a matrix exponential more efficiently than the algorithm using diagonalization.

Next, we will discuss an efficient algorithm for eigenvalues for this matrix in the next section.

3. Eigenvalues Using Viète's Formula

In this section, we investigate an algorithm using Viète's formula. This algorithm achieves a fast computation of eigenvalues in comparison with the algorithm using diagonalization.

If $S \in$ Sym(3) is given, we need to solve the characteristic

equation of S. Let λ_1, λ_2 and λ_3 be the eigenvalues of S. Then, the characteristic equation is $(x-\lambda_1)(x-\lambda_2)(x-\lambda_3) = 0$. This equation is expanded into

$$x^3 - (\lambda_1+\lambda_2+\lambda_3)x^2 + (\lambda_1\lambda_2+\lambda_2\lambda_3+\lambda_3\lambda_1)x - \lambda_1\lambda_2\lambda_3 = 0.$$

Therefore, the characteristic equation of S is described as

$$x^3 - (\mathrm{tr}S)x^2 + \frac{(\mathrm{tr}S)^2 - \|S\|_F^2}{2}x - \det S = 0,$$

where $\mathrm{tr}S$ is the trace of S, $\|S\|_F$ is the Frobenius norm of S.

This equation can be solved by using Viète's formula (see Appendix A.1) . As a result, we obtain the eigenvalues of S as follows.

$$\lambda_1 = \frac{r\cos\left(\frac{\arccos(k)}{3}\right) + \mathrm{tr}S}{3},$$
$$\lambda_2 = \frac{r\cos\left(\frac{\arccos(k)+2\pi}{3}\right) + \mathrm{tr}S}{3},$$
$$\lambda_3 = \frac{r\cos\left(\frac{\arccos(k)+4\pi}{3}\right) + \mathrm{tr}S}{3},$$

where

$$k = -\frac{108q}{r^3},\ r = \sqrt{-12p},$$
$$p = \frac{(\mathrm{tr}S)^2 - 3\|S\|_F^2}{6},\ q = \frac{5}{54}(\mathrm{tr}S)^3 - \frac{1}{6}\mathrm{tr}S\,\|S\|_F^2 - \det S.$$

We note that the eigenvalues of S can be represented by a simple formula using the trace, determinant and Frobenius norm of S. The proposed method for computing eigenvalues involves computation of cosine and arccosine. Cosine is implemented in the Eigen library. The function comes from Julien Pommier's SSE math library which is known as an accurate math library, and arccosine is implemented in the C++ Standard Library. We consider the speed and accuracy of these functions are acceptable.

4. Applications

We incorporated our implementation of the algorithm discussed above into the shape deforming tools developed in Ref. [6]. For example, Cage-based deformer gives a target shape and a "cage" surrounding it. The cage can be any triangulated polyhedron wrapping the target shape. We want to deform the target shape by manipulating not directly on it but through the proxy cage. Essentially this deformation is based on polar decompositions and matrix exponentials of 3×3 real symmetric matrices. We examined our improvements replacing the exponential part in this deformation program.

5. Experimentation

In this section, we compared the computation times of our algorithms with standard techniques. We use a Intel Core i7 1.9 GHz CPU with 4 GB of RAM and Windows 8 (64 bit) OS. In the experimentations, we implemented our algorithms in C++ programming language (Microsoft Visual Studio 2010 Ultimate), and the optimization option (/O2) of this compiler was maximization of the execution speed. Our implementation used the Eigen library which provides many functions of matrix operations [4].

First we compared our algorithm for the computation of the eigenvalues of a 3×3 real symmetric matrix with the QR algorithm [11] which uses the QR decomposition. This algorithm is implemented in the Eigen library as a class member function which computes the eigenvalues. Before call the function, we need to create an instance of this class by specifying the size of matrix. Then, the QR algorithm is restricted to 3×3 matrix. For given 10^7 three dimensional real symmetric matrices randomly, we compared the running times for the computation of eigenvalues of all given matrices. In the applications of CG, the eigenvalues of the given matrices should be more than 0. Hence, we need to obtain positive semi-definite matrices. The way of generating these matrices randomly is the following. First, we generate a real matrix M and choose the elements of M from the range $[-1, 1]$. Next, we compute $S = {}^tMM$ to obtain a real symmetric matrix. Then, the eigenvalues of S are more than 0 because S is a positive semi-definite matrix. **Figure 1** shows a comparison of the computation of the eigenvalues using the QR algorithm and our algorithm using Viète's formula. Our algorithm is about 69% faster than the QR algorithm.

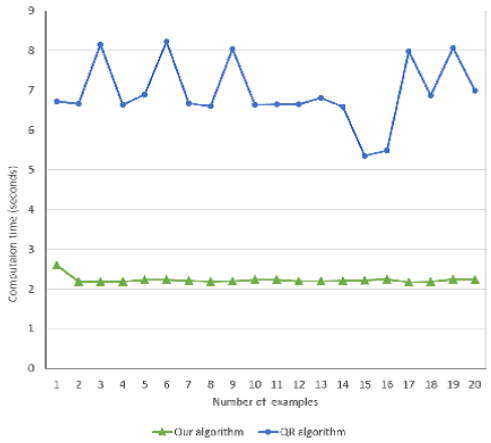


Fig. 1 Computation time of the eigenvalues of the 3×3 real symmetric matrix. Upper: using the QR algorithm. Lower: using our algorithm.

The Eigen library provides an algorithm computing the eigenvalues using Viète's formula. However, we implemented our algorithm independently using our formalized formula using the trace, determinant and Frobenius norm.

Next, we compared our algorithm for the computation of the matrix exponential of a 3×3 real symmetric matrix with an algorithm based on diagonalization. For given 10^7 three dimensional real symmetric matrices chosen randomly by the same method in the previous experimentation, we compared the average running times for computing the matrix exponentials using the algorithm based on diagonalization (Diag), spectral decomposition (SD) and improved spectral decomposition (improved SD) which includes our algorithm for the computation of the eigenvalues. **Figure 2** shows a comparison of the computation of the matrix exponentials using the three algorithms. As a result, improved SD is the best algorithm among those algorithms and achieved about 76% faster than Diag.

We consider eigenvalues between 0 and 3, because a transformation matrix induced by an interactive motion do not have large eigenvalues. By our experiments, the error rate is less than 10^{-6} and it is acceptable for making a deformation of computer

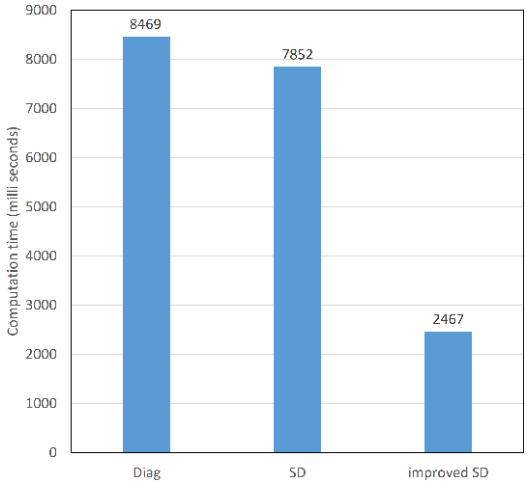


Fig. 2 A comparison of average runtimes for the computation of the matrix exponential of 3×3. Left: using diagonalization (Diag). Center: using spectral decomposition (SD). Right: using the improved spectral decomposition (improved SD) which includes our algorithm for the computation of the eigenvalues.

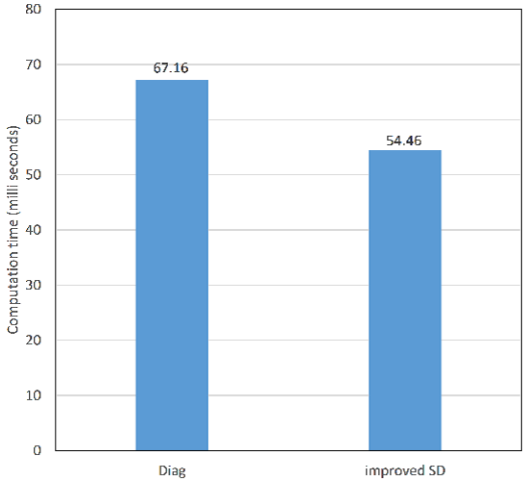


Fig. 3 A comparison of average runtimes for the deformation of a dragon model. Left: using diagonalization (Diag). Right: using improved spectral decomposition (improved SD) which includes our algorithm for the computation of the eigenvalues.

graphics. Our method using Viète's formula may produce small numerical errors, but we have benefits of computing time using our exponential matrix without computing eigenvectors. Since we consider $\lambda \leq 3$, the error rate of the difference of cases in our algorithm are bounded. Experimentally, the computation of deformations are acceptable.

Finally we incorporated the implementations of Diag and improved SD into the Cage-based deformer algorithm in Ref. [6]. For a dragon model with 50,000 vertices, 100,000 triangles, we compared the average running times of deforming this model using two algorithms. **Figure 3** shows the comparison of the computation times, and the average runtime of improved SD is about 19% faster than Diag. Compared with the result for the eigenvalues, this improvement in speed for the application may not look like significant. This is because the application has other heavy computations. In more detail, the application computes affine transformations for all triangular meshes of a given model and logarithms of matrices obtained from the affine transformations. In addition, the application needs a weighted sum calculation to decide the shape deformation.

Therefore, our algorithm can be useful for a practical deformation tool.

6. Conclusion

We investigate fast algorithms for the matrix exponentials and eigenvalues of 3×3 real symmetric matrices. We show that the matrix exponential and eigenvalues of the 3×3 real symmetric matrix can be represented by specific formulas using the trace, determinant and Frobenius norm of the matrix. Our algorithms are implemented in C++ programming language as functions which compute the matrix exponential and eigenvalues of the 3×3 real symmetric matrix. For the computation time of the matrix exponential, we compare our algorithm with other algorithms. In addition, we incorporate the implementations into a practical deformation tool. Using the deformation tool, we evaluate the computation time of our algorithm for the deformation of a given model.

In conclusion, we achieve a notable performance improvement using our algorithm. In fact, we compare the computation of the eigenvalues using the QR algorithm and our algorithm using Viète's formula. Our algorithm is about 69% faster than the QR algorithm. Next, we compare the computation of the matrix exponentials using an algorithm based on diagonalization, our algorithm based on spectral decomposition and our improved algorithm which includes our algorithm computing the eigenvalues using Viète's formula. Then, our improved algorithm is the best algorithm among those algorithms and achieved about 76% faster than the algorithm using diagonalization. Finally we incorporate the implementations of the algorithm using diagonalization and our improved algorithm into the Cage-based deformer. For a dragon model, we compare average running times of deforming this model using two algorithms. As a result, the average runtime of our improved algorithm is about 19% faster than the algorithm based on the diagonalizaion.

We have not estimated the accurate numerical error of our matrix exponential algorithm. In this paper, we just show that the benefits of computational costs and acceptable results of applications of deformations. Future work includes the investigation of numerical errors and their effects for some specific deformations.

Acknowledgments We would like to express our gratitude to K. Anjyo, H. Ochiai, S. Kaji and Y. Mizoguchi for their valuable suggestions and encouragements. We also would like to thank S. Hirose, S. Yokoyama and G. Matsuda for useful discussions and comments. This work is partially supported by Kyushu University Global COE Program "Education-and-Research Hub for Mathematics-for-Industry" and Core Research for Evolutional Science and Technology (CREST) Program "Mathematics for Computer Graphics" of Japan Science and Technology Agency (JST).

References

[1] Alexa, M.: Linear Combination of Transformations, *Proc. 29th Annual Conference on Computer Graphics and Interactive Techniques*, pp.380–387, ACM (2002).

[2] Alexa, M., Cohen-Or, D. and Levin, D.: As-rigid-as-possible Shape Interpolation, *Proc. 27th Annual Conference on Computer Graphics*

and Interactive Techniques, pp.157–164, ACM Press/Addison-Wesley Publishing Co. (2000).
[3] Brockett, R.: Robotic Manipulators and the Product of Exponentials Formula, *Proc. MTNS-83 International Symposium*, Fuhrmann, P., (Ed.),pp.120–129, Springer Berlin Heidelberg (1983).
[4] Guennebaud, G., Jacob, B., et al.: Eigen v3 (online), available from ⟨http://eigen.tuxfamily.org⟩ (accessed 2014-6-25).
[5] Igarashi, T. and Igarashi, Y.: Implementing As-Rigid-As-Possible Shape Manipulation and Surface Flattening, *J. Graphics, GPU, & Game Tools*, Vol.14, No.1, pp.17–30 (2009).
[6] Kaji, S., Hirose, S., Ochiai, H. and Anjyo, K.: A Lie Theoretic Parameterization of Affine Transformations, *Proc. Symposium MEIS2013: Mathematical Progress in Expressive Image Synthesis*, MI Lecture Note, Vol.50, pp.134–140, Kyushu University (2013).
[7] Kaji, S., Hirose, S., Sakata, S., Mizoguchi, Y. and Anjyo, K.: Mathematical Analysis on Affine Maps for 2D Shape Interpolation, *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, pp.71–76 (2012).
[8] Matsushita, K., Hamada, H. and Matsuda, G.: A Fast Computation of Matrix Exponential and its Application in CG, *Proc. Forum "Math-for-Industry" 2013*, MI Lecture Note, Vol.51, p.92, Kyushu University (2013).
[9] Moler, C. and Loan, C.V.: Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later, *SIAM Review*, Vol.45, No.1, pp.3–49 (2003).
[10] Sumner, R.W., Zwicker, M., Gotsman, C. and Popović, J.: Mesh-based Inverse Kinematics, *ACM Trans. Graph.*, Vol.24, No.3, pp.488–495 (2005).
[11] Wilkinson, J.H. (Ed.): *The Algebraic Eigenvalue Problem*, Oxford University Press, Inc. (1988).

Appendix

A.1 Computing the Eigenvalues Using Viète's Formula

For a given $S \in \mathrm{Sym}(3)$, let λ_1, λ_2 and λ_3 be the eigenvalues of S. Those are the roots of the characteristic equation of S:

$$x^3 - (\mathrm{tr}S)x^2 + \frac{(\mathrm{tr}S)^2 - \|S\|_F^2}{2}x - \det S = 0.$$

Let a, b, c, and y be $a = -\mathrm{tr}S$, $b = \frac{(\mathrm{tr}S)^2 - \|S\|_F^2}{2}$, $c = -\det S$ and $y = x + \frac{a}{3}$.

Then,

$$y^3 + \left(b - \frac{1}{3}a^2\right)y + \left(\frac{2}{27}a^3 - \frac{1}{3}ab + c\right) = 0.$$

Let p and q be $p = b - \frac{1}{3}a^2$ and $q = \frac{2}{27}a^3 - \frac{1}{3}ab + c$. Since all eigenvalues of real symmetric matrices are real numbers, the solutions are also real numbers. Hence, the discriminant satisfies $-(4p^3 + 27q^2) \geq 0$, especially $p \leq 0$. If $p = 0$, then we find $q = 0$. Then, λ_1, λ_2 and λ_3 are equal to $\frac{\mathrm{tr}S}{3}$. So we consider the case of $p < 0$. Let $t = \sqrt{-\frac{4p}{3}}$, $u = \frac{y}{t}$ and $k = -\frac{4q}{t^3}$. Then, the equation is described as

$$4u^3 - 3u - k = 0.$$

Since $4p^3 + 27q^2 \leq 0$, we have $|q| \leq \sqrt{-4p^3/27}$, $|4q| \leq t^3$ and $|k| \leq 1$. To solve the equation, we use the cosine's Triple-angle formula $\cos 3\theta = 4\cos^3\theta - 3\cos\theta$. Let u_1, u_2 and u_3 be the roots of the equation, and set $\theta = \frac{1}{3}\arccos(k)$. From this formula, the roots are

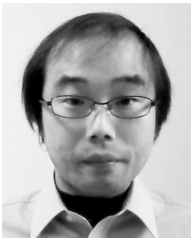
$$u_1 = \cos\left(\frac{1}{3}\arccos k\right),$$
$$u_2 = \cos\left(\frac{1}{3}\arccos k + \frac{2}{3}\pi\right),$$
$$u_3 = \cos\left(\frac{1}{3}\arccos k + \frac{4}{3}\pi\right).$$

From $y = x + \frac{a}{3}$, $u = \frac{y}{t}$, the eigenvalues of S are $\lambda_i = tu_i - \frac{a}{3}$ $(i = 1, 2, 3)$. Therefore, we find a simple formula for the eigenvalues of a 3×3 symmetric matrix.

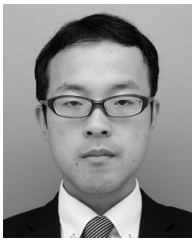
$$\lambda_1 = \frac{r\cos\left(\frac{\arccos(k)}{3}\right) + \mathrm{tr}S}{3},$$
$$\lambda_2 = \frac{r\cos\left(\frac{\arccos(k)+2\pi}{3}\right) + \mathrm{tr}S}{3},$$
$$\lambda_3 = \frac{r\cos\left(\frac{\arccos(k)+4\pi}{3}\right) + \mathrm{tr}S}{3},$$

where

$$r = 3t = \sqrt{-12p},\ k = -\frac{108q}{r^3},$$
$$p = \frac{(\mathrm{tr}S)^2 - 3\|S\|_F^2}{6},\ q = \frac{5}{54}(\mathrm{tr}S)^3 - \frac{1}{6}\mathrm{tr}S\,\|S\|_F^2 - \det S.$$



Kohei Matsushita received his B.S. and M.M. degrees from Kyushu University in 2010 and 2012. He is a doctoral student in the Graduate School of Mathematics, Kyushu University. His research interests are numerical linear algebra for computer graphics and optimization techniques. He is a member of the IPSJ.



Hiroyasu Hamada received his M.M. and Ph.D degrees from Kyushu University in 2008 and 2012. He is a lecturer at National Institute of Technology, Sasebo College. His research interests are functional analysis, especially operator algebras, and numerical linear algebra for computer graphics. He is a member of the MSJ.