1994

Numerical Determination of Tristimulus Values

Borges, C.F.

C.F. Borges, Numerical Determination of Tristimulus Values, Journal of the Optical Society of America A, Vol. 11, No. 12, December 1994, pp. 3152-3161.
https://hdl.handle.net/10945/38062

Numerical determination of tristimulus values

Carlos F. Borges

Code Ma/Bc, Naval Postgraduate School, Monterey, California 93943

Received February 14, 1994; revised manuscript received June 13, 1994; accepted July 18, 1994

I consider numerical methods for evaluating the tristimulus values of P_λ, an arbitrary spectral power distribution. Various classical quadrature rules are discussed, and numerically stable algorithms for their construction are presented. I introduce a new quadrature rule developed specifically for evaluating tristimulus values. The method involves the simultaneous generation of three quadrature rules sharing a common set of nodes that is optimal in a sense much like the optimality of the Gauss rules.

1. INTRODUCTION

A common problem in colorimetry and colorimetric computations is to compute the tristimulus values of a given spectral power distribution P_λ. Under certain conditions one can apply Grassmann's laws and reduce this problem to that of evaluating the following set of definite integrals:

$$X = \int_\nu P_\lambda \bar{\mathbf{x}}(\lambda)\mathrm{d}\lambda,$$
$$Y = \int_\nu P_\lambda \bar{\mathbf{y}}(\lambda)\mathrm{d}\lambda,$$
$$Z = \int_\nu P_\lambda \bar{\mathbf{z}}(\lambda)\mathrm{d}\lambda, \qquad (1)$$

where ν is the visible interval and $\bar{\mathbf{x}}(\lambda)$, $\bar{\mathbf{y}}(\lambda)$, and $\bar{\mathbf{z}}(\lambda)$ are the color-matching functions.[1] This paper is concerned with constructing effective techniques for evaluating these integrals.

To begin, note that since the color-matching functions are empirically determined through color-matching experiments and are not known analytically (although some analytic approximations have been given),[2] it is common practice to evaluate the definite integrals in Eqs. (1) numerically. The most widely used methods are based on simple numerical quadrature schemes such as the composite-rectangle rule. There has been little research into the use of more-sophisticated techniques. Wallis[3] gives the weights and nodes for applying Gauss quadrature rules as high as order 6 to this problem, but he does not give stable algorithms for generating them. Such algorithms are necessary if one chooses to use color spaces or bias illuminants other than the ones given in his paper. In this paper I discuss stable methods for the construction of interpolatory and Gauss quadrature rules for any color space. I also discuss a new and powerful method for constructing a trio of rules that share a common set of nodes. These rules have some remarkable properties that are especially attractive for this problem.

2. MOTIVATION

Numerical estimation of colorimetric integrals is an important problem and appears in many applications such as image processing, computer vision, and computer graphics (image synthesis). Often it is necessary to evaluate these integrals millions of times rapidly or to estimate them to high accuracy with a minimal number of spectral samples. This paper will examine the construction of primitive quadrature rules (i.e., weighted sums) for estimating colorimetric integrals. Rules of this form are important because they can give high-accuracy estimates of colorimetric integrals with a small number of spectral samples and hence can be computed rapidly. For example, if one can achieve the desired accuracy by using only 10 spectral samples, in contrast to the 471 spectral samples required for summation at 1-nm intervals from 360 to 830 nm, then one can compute 47 times faster by using the ten-point rule.

Three different situations will be examined. Section 3 addresses the situation in which one can select the spectral points at which the data will be taken and in which one wishes to construct rules that give as much accuracy as possible from each estimate. This scheme leads to Gauss quadrature rules of the form given by Wallis in Ref. 3. Section 4 addresses the situation in which one is given an arbitrary set of spectral points at which data are available and in which one wishes to construct rules that use this data to estimate the integrals. This leads to interpolatory quadrature rules. Section 5 involves a variation of the first case, in which one can choose the spectral points but with the restriction that the same spectral points be used to compute each of the three integrals. This leads to a new class of quadrature rules that I call quasi-Gaussian.

It is important to emphasize the goal of this paper. In short, I wish to demonstrate how one should go about constructing quadrature rules for estimating colorimetric integrals. All the rules considered in this paper take the form of weighted sums with a small number of terms; hence they are simple and fast in application. However, constructing these rules can be delicate and involved. This is the process that I wish to address. Once such a rule has been constructed, the rest is easy.

It is also important to note what is not addressed in this paper. In particular, I assume throughout that the spectral power density (or reflectance, transmittance, etc.) can be accurately sampled with small bandpass (1 nm). The effects of larger sample bandpass on the calculation of tristimulus values is in itself a formidable question[4,5] and will not be addressed here.

0740-3232/94/123152-10$06.00

3. GAUSS RULES FOR COLOR MATCHING

Wallis[3] gives the weights and nodes of the Gauss quadrature rules as high as order 6 for computing the XYZ values of spectral power density and also for computing the XYZ values of a surface under various types of illumination. His motivation was that he could get high accuracy with a small number of spectral samples and hence could greatly increase the speed of colorimetric computations. These rules generally require that different spectral samples be used in the estimation of each of the three colorimetric integrals. I begin with a brief review of these rules and their construction.

The Gauss quadrature rule of order n associated with the weight function $\omega(x)$ is an n-point primitive rule; i.e., it has the form

$$\int_a^b f(x)\omega(x)\mathrm{d}x \approx \sum_{i=1}^{n} f(x_i)w_i\,. \tag{2}$$

The x_i are called the Gauss abscissas (or nodes), and the w_i are called the Gauss weights. Gauss rules are powerful because they have degree of precision $2n-1$, where the degree of precision is defined to be the largest integer d such that

$$\int_a^b p(x)\omega(x)\mathrm{d}x \approx \sum_{i=1}^{n} p(x_i)w_i \tag{3}$$

for all polynomials $p(x)$ of degree d or less. In general, no n-point primitive rule has a higher degree of precision that the corresponding Gauss rule. This follows from the fact that there are only $2n$ degrees of freedom in constructing an n-point primitive rule (the choice of n nodes and n weights), and a Gauss rule is the unique set of weights and nodes that maximizes the degree of precision.

Gauss rules are intimately connected to the orthogonal polynomial system associated with the weight function $\omega(x)$ (see Ref. 6). The orthogonal polynomial system associated with $\omega(x)$ is a set of polynomials $\{p_n(x)\}_{n=0}^{\infty}$, where $p_j(x)$ is a polynomial of exact degree j and

$$\int_a^b p_i(x)p_j(x)\omega(x)\mathrm{d}x = 0 \qquad \text{if } i \neq j\,.$$

The Gauss abscissas are the roots of the nth orthogonal polynomial associated with $\omega(x)$, and the Gauss weights can be determined with the same orthogonal polynomials. It can be shown that the Gauss abscissas are real and distinct and lie in the interval of integration, provided that $\omega(x)$ is nonnegative in this interval; weight functions that satisfy this criterion are called admissible.

The procedure that Wallis describes for constructing Gauss rules[3] is based on this relationship and on the fact that the orthogonal polynomial system asociated with $\omega(x)$ on $[a, b]$ satisfies a three-term recurrence relation of the form

$$p_{n+1}(x) = (x-\alpha_n)p_n(x) - \beta_n{}^2 p_{n-1}(x) \tag{4}$$

for $n = 0, 1, 2, \ldots$, where

$$p_{-1}(x) = 0, \qquad p_0(x) = 1\,.$$

It is not hard to show that the recurrence coefficients are given by

$$\alpha_n = \frac{\int_a^b x p_n{}^2(x)\omega(x)\mathrm{d}x}{\int_a^b p_n{}^2(x)\omega(x)\mathrm{d}x}, \qquad n = 0, 1, 2, \ldots,$$

$$\beta_n{}^2 = \frac{\int_a^b p_n{}^2(x)\omega(x)\mathrm{d}x}{\int_a^b p_{n-1}{}^2(x)\omega(x)\mathrm{d}x}, \qquad n = 1, 2, 3, \ldots, \tag{5}$$

and $\beta_0{}^2 = \int_a^b \omega(x)\mathrm{d}x$. It is clear from the second equation in Eqs. (5) that the β_n determine the normalization factors, since $\int_a^b p_n{}^2(x)\omega(x)\mathrm{d}x = \beta_0{}^2\beta_1{}^2\ldots\beta_n{}^2$.

Alternately applying the two equations in Eqs. (5), one can build up the system of monic orthogonal polynomials. This process is known as the Stieltjes procedure, and Wallis[3] uses it to construct the nth orthogonal polynomial from which he then extracts the roots (Gauss abscissas), using a root finder, and the Gauss weights, using the equation

$$w_i = \frac{1}{p_n'}(x_i)\int_a^b \frac{p_n(x)}{x-x_i}\omega(x)\mathrm{d}x\,. \tag{6}$$

Wallis points out that the Stieltjes procedure is better than the classical Gram–Schmidt procedure, but he does not point out that the Stieltjes procedure is itself unstable. Color matching is even more delicate, because the absence of analytic expressions for the matching functions introduces additional errors into the computation of the recurrence coefficients by means of Eq. (5). Further errors are introduced when one is solving for the roots, and these errors propagate into the calculation of the weights.

One can see the limitations of this method by examining the weights given in Ref. 3 for the various rules. By definition, each of the XYZ matching functions has unit 1 norm, that is,

$$\int_\nu \bar{\mathbf{x}}(\lambda)\mathrm{d}\lambda = \int_\nu \bar{\mathbf{y}}(\lambda)\mathrm{d}\lambda = \int_\nu \bar{\mathbf{z}}(\lambda)\mathrm{d}\lambda = 1\,.$$

Hence the properties of Gauss quadrature imply that the weights of Wallis's quadrature rules must add up to 1. This is not true of the weights given in Ref. 3, as one can verify by looking at Table 1. Twelve rules are presented, and only half satisfy this criterion to the precision shown, five are correct to only three significant digits, and one is correct to only one significant digit (it seems reasonable to assume that this is a typographical error, but it serves to emphasize the need for presenting workable algorithms rather than just tabulated data). Fortunately, there are more stable ways of constructing Gauss rules.

The first step is to use a method for finding the nodes and weights that does not require explicit construction of the orthogonal polynomials. One such method is the Golub–Welsch algorithm,[7] which uses the three-term recurrence coefficients directly to construct the rule. The method uses the fact that the three-term recurrence relation [Eq. (4)] can be rewritten in the following matrix form (see Wilf[8]):

$$x\mathbf{p}_n = J_n\mathbf{p}_n + \beta_n\mathbf{e}_n p_n(x)\,, \tag{7}$$

Table 1. Gauss Rules for CIE *XYZ* Matching Functions Biased with CIE Standard Illuminant A

	X		*Y*		*Z*	
Order	λ	Weight	λ	Weight	λ	Weight
3	453.6	0.065261	0.503874	0.177165	0.431958	0.127213
	585.4	0.730193	0.573064	0.663361	0.474276	0.213467
	650.5	0.303221	0.644716	0.159474	0.540278	0.015236
4	440.7	0.043219	0.478896	0.047905	0.421835	0.055088
	553.5	0.297236	0.544182	0.489976	0.457523	0.225467
	614.9	0.687473	0.606651	0.428121	0.502893	0.073651
	678.7	0.070747	0.674579	0.033998	0.579973	0.001709
5	433.4	0.030186	0.459534	0.013939	0.410887	0.016586
	500.2	0.063214	0.523022	0.275672	0.443499	0.169740
	584.1	0.607578	0.578274	0.527233	0.479739	0.150248
	638.4	0.385966	0.635738	0.178715	0.528279	0.018902
	708.2	0.011731	0.707747	0.004441	0.601692	0.000441
6	423.6	0.014005	0.445510	0.004843	0.399538	0.003869
	462.8	0.042873	0.504633	0.126493	0.432846	0.103662
	562.4	0.326747	0.554856	0.461960	0.464566	0.186371
	613.3	0.596717	0.607336	0.354234	0.503312	0.058073
	664.8	0.117059	0.662150	0.051969	0.555295	0.003767
	741.7	0.001273	0.740704	0.000500	0.613557	0.000175

where

$$y_n = \begin{bmatrix} \alpha_0 & \beta_1 & & \\ \beta_1 & \alpha_1 & \beta_2 & \\ & \beta_2 & \alpha_2 & \ddots \\ & & \ddots & \ddots \end{bmatrix} \tag{8}$$

is the $n \times n$ Jacobi matrix associated with $w(x)$ and the α_i and β_i are those given by Eqs. (5), $\mathbf{p}_n{}^T = [p_0(x), p_1(x), \ldots, p_{n-1}(x)]$ is a vector of the first n orthogonal polynomials associated with $w(x)$, and $\mathbf{e}_n{}^T = [0, 0, \ldots, 0, 1]$ is the nth axis vector.

It is clear from Eq. (7) that the roots of the orthogonal polynomial of order n are the eigenvalues of the Jacobi matrix J_n. Moreover, it is known that the weights of the nth order Gauss rule for a normalized weight function (i.e., $\beta_0 = 1$) are the squared first elements of the normalized eigenvectors of J_n. If $\beta_0 \neq 1$, then the weights are found by multiplication of the squared first elements of the normalized eigenvectors by β_0. The Golub–Welsch algorithm exploits this relationship by approaching the construction of a Gauss rule as an eigenvalue problem. In particular, it is suggested that the best way to construct a Gauss rule is to apply the QR algorithm of Francis[9] to the Jacobi matrix J_n. The outputs of this stable algorithm (or one of its variants for symmetric tridiagonal matrices) are the eigenvalues and normalized eigenvectors that can be used to construct the Gauss rule directly.

All that remains is to find a stable method for constructing the Jacobi matrix. For some weight functions the recurrence coefficients, and hence the Jacobi matrices, are known explicitly. This is not the case for color matching. There is, fortunately, a method due to Gautschi[10] called the discretized Stieltjes procedure, which allows us to generate them in a stable manner. This approach replaces the continuous inner product

$$\langle f(x), g(x)\rangle_\omega = \int_a^b f(x)g(x)w(x)\mathrm{d}x \tag{9}$$

from Eqs. (5) with a sequence of discrete inner products,

$$\langle f(x), g(x)\rangle_{\omega^{(k)}} = \sum_{i=1}^{k} f[x_i^{(k)}]g[x_i^{(k)}]w[x_i^{(k)}], \tag{10}$$

that converge to the continuous one. This sequence of discrete inner products is used to create a sequence of matrices, which can be shown to converge to the Jacobi matrix for the given weight function. The details of this method are outlined in Ref. 11, and I will not pursue them here other than to say that the method is easily implemented and applied to the color-matching problem. A somewhat more stable and efficient variant can be adapted from the Gragg–Harrod algorithm.[12]

Gauss rules for any matching function can be constructed with the following algorithm:

Algorithm 1: Let n be the order of the desired Gauss rule, $\omega(\lambda)$ be the matching function, and P_λ be the illuminant bias (with $P_\lambda \equiv 1$ if no illuminant bias is desired).

1. Use the discretized Stieltjes procedure with weight function $P_\lambda\omega(\lambda)$ to construct J_n, the nth-order Jacobi matrix associated with this illuminant-matching function pair.
2. Use the symmetric QR algorithm to find the eigenvalues and the normalized eigenvectors of J_n.
3. The Gauss abscissas λ_i are the eigenvalues from the step 2, and the Gauss weights w_i are the squared first elements of the associated normalized eigenvectors weighted by

$$\int_\nu P_\lambda\omega(\lambda)\mathrm{d}\lambda.$$

It will be useful to introduce the notion of a performance ratio for a set of quadrature rules, defined as

$$R = \frac{\text{Overall degree of precision}_{+1}}{\text{Number of integrand evaluations}}, \tag{12}$$

where the overall degree of precision for a set of rules is

defined to be the largest integer such that every rule in the set gives at least that much precision. This ratio indicates how many overall degrees of precision are yielded by each integrand evaluation. It follows that a set of three n-point Gauss rules has performance ratio

$$R = \frac{(2n-1)+1}{3n} = \frac{2}{3}.$$

As we shall see, this indicates rather poor performance.

4. CONSTRUCTING INTERPOLATORY RULES

An interpolatory quadrature rule is one in which the nodes are chosen arbitrarily and the weights are selected in a way that maximizes the polynomial degree of precision. For colorimetric computations these rules are appropriate when one is given the points at which the spectral samples are taken and must use these data. Conceptually, the simplest way in which to construct an interpolatory rule is to choose the weights so that the rule correctly generates the first n moments of the associated weight function. In other words, it is required that

$$\sum_{i=1}^{n} {x_i}^j w_i = \int_a^b x^j \omega(x)\mathrm{d}x \tag{11}$$

for $j = 0, 1, 2, \ldots, n-1$. This gives a set of n linear equations in the unknown weights $\{w_i\}_{i=1}^n$. In matrix form

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ {x_1}^2 & {x_2}^2 & \cdots & {x_n}^2 \\ \vdots & \vdots & & \vdots \\ {x_1}^{n-1} & {x_2}^{n-1} & \cdots & {x_n}^{n-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{n-1} \end{bmatrix}, \tag{12}$$

where

$$\mu_i = \int_a^b x^i \omega(x)\mathrm{d}x.$$

Note that the matrix is Vandermonde; there are a number of powerful methods for solving these systems (see Refs. 13 and 14). Although this approach works, it is well documented that methods that rely on moment sequences are inherently unstable (for examples, see Ref. 10). A different approach is presented in Ref. 15, although in a general form. Since I do not consider the case in which derivative data about the integrand are available, the algorithm can be considerably simplified. Here is a brief explanation.

The algorithm relies on the fact that interpolatory quadrature on a given set of nodes is equivalent to constructing the interpolating polynomial $p(x)$ such that $p(x_i) = f(x_i)$ for $i = 1, 2, \ldots, n$ and then computing

$$\int_a^b p(x)\omega(x)\mathrm{d}x$$

exactly.

If no derivative data are used and the nodes x_i are distinct, then $p(x)$ is the Lagrange interpolating polynomial,[16]

$$p(x) = \sum_{i=1}^{n} f(x_i) l_i(x),$$

where

$$l_i(x) = \frac{(x-x_1)(x-x_2)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_1)(x_i-x_2)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}.$$

Integrating yields

$$\sum_{i=1}^{n} f(x_i) \int_a^b l_i(x)\omega(x)\mathrm{d}x,$$

and it follows that the interpolatory weights, w_i, are given by

$$w_i = \int_a^b l_i(x)\omega(x)x\mathrm{d}.$$

Since the integrands in Eq. (4) are polynomials of degree $n-1$, the definite integrals in Eq. (4) can be computed exactly by use of the Gauss rule of order $\lfloor n + 1/2 \rfloor$, where $\lfloor x \rfloor$ is the largest integer less than x.

Kautsky and Elhay[15] claim that this approach is noticeably more stable than is solving the Vandermonde system in the moments. Moreover, since it was shown in Section 3 how to construct Gauss rules for color matching, it is straightforward to compute the weights of an interpolatory rule in this way. Here is the algorithm:

Algorithm 2: Let n be the order of the desired interpolatory rule, $\omega(\lambda)$ be the matching function, P_λ be the illuminant bias (with $P_\lambda \equiv 1$ if no illuminant bias is desired), and $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the nodes.

1. Use algorithm 1 to construct the Gauss rule of order $k = \lfloor n + 1/2 \rfloor$ for the given illuminant-matching function pair. This yields a set of k Gauss abscissas λ_i and Gauss weights $\hat{w}_i$.
2. Compute the interpolatory weights w_i, using

$$w_i = \sum_{j=1}^{k} \hat{w}_j l_i(\hat{\lambda}_j).$$

The algorithm is fast and easy to implement. But, most important, it has good numerical stability because it does not rely on the moments. Note that if some of the interpolatory nodes are also Gauss nodes then some of the terms in the above summation will vanish.

If we construct three interpolatory rules based on a single set of n distinct nodes then the performance ratio is

$$R = \frac{(n-1)+1}{n} = 1,$$

which is better than that for a set of three Gauss rules.

5. CONSTRUCTING OPTIMAL SHARED-NODE RULES

In this section I show how to construct an optimal set of shared-node rules developed especially for the numerical

determination of tristimulus values. The main attraction of these rules is that they achieve the highest overall accuracy with the fewest number of spectral samples. These rules are appropriate when one can choose the spectral sampling points but must use the same ones for all three integrals. The development proceeds by mimicking the development of Gauss quadrature rules.

Derivation of the Gauss rules amounts to demanding a set of n nodes and n weights that maximize the degree of precision for a single weight function. In Section 3 we were solving for three distinct n-point Gauss rules with a total of $6n$ unknowns: n nodes and n weights for each of the three matching functions. However, for a set of three shared-node rules there are only $4n$ unknowns (n weights for each of the three matching functions, plus n nodes). It seems reasonable to use $4n$ equations and demand that the nodes and weights maximize the overall degree of precision. Assume that $n = 3l$ (the restriction that n be a multiple of 3 is artificial and can be removed to produce a more general derivation; see Ref. 16 and consider the following method):

Let each $\tilde{\mathbf{Q}}_k$ be of the form

$$\tilde{\mathbf{Q}}_k f = \sum_{i=1}^{n} f(\lambda_i) w_{k,i},$$

where the weights $w_{k,i}$ and the nodes $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ satisfy the following $4n$ equations:

$$\begin{aligned}
\sum_{i=1}^{n} w_{k,i} &= \int_a^b \omega_k(\lambda)\mathrm{d}\lambda, \\
\sum_{i=1}^{n} \lambda_i w_{k,i} &= \int_a^b \lambda\omega_k(\lambda)\mathrm{d}\lambda, \\
\sum_{i=1}^{n} \lambda_i{}^2 w_{k,i} &= \int_a^b \lambda^2\omega_k(\lambda)\mathrm{d}\lambda, \\
&\vdots \\
\sum_{i=1}^{n} \lambda_i{}^{n+l-1} w_{k,i} &= \int_a^b \lambda^{n+l-1}\omega_k(\lambda)\mathrm{d}\lambda,
\end{aligned} \tag{13}$$

for $k = 1, 2, 3$.

Each rule will have degree of precision of $n + l - 1$, and the overall degree of precision will be $n + l - 1$, which is, in general, the maximum achievable. The performance ratio of this set of rules is $R = 4/3$.

The equations in Eqs. (13) are nonlinear but can be solved with a variant of Prony's method, which is straightforward but not numerically stable. It is better to develop a method more of the nature of the Golub–Welsch algorithm, in particular, one that uses the properties of Jacobi matrices and orthogonal polynomials.

In a sense the optimal shared-node rules trade precision for relaxed orthogonality conditions. Whereas an n-point Gauss rule requires that the nodes be the roots of the nth orthogonal polynomial associated with the weight function, the shared-node rules do something more subtle. Let $q(\lambda)$ be an nth-degree polynomial whose roots are the nodes of our rule; then $q(\lambda)$ will be orthogonal to all polynomials of degree less than l with respect to each matching function. It can be shown[16] that $q(\lambda)$ can be written as

$$q(\lambda) = \sum_{i=l}^{n} \gamma_i p_i{}^*(\lambda), \tag{14}$$

where $p_i{}^*(\lambda)$ is the ith element of the orthogonal polynomial system associated with $\omega_1(\lambda)$. The remaining orthogonality conditions imply that $q(\lambda)$ must also satisfy that

$$\int_\nu q(\lambda) p_j{}^*(\lambda)\omega_k(\lambda)\mathrm{d}\lambda = 0 \tag{15}$$

for $j = 0, 1, \ldots, l - 1$ and $k = 2, 3$. Henceforth let $\mu_i^{(k)}$ denote the ith modified moment for the kth weight function,

$$\mu_{i,j}^{(k)} = \int_\nu p_i{}^*(\lambda)\omega_k(\lambda)\mathrm{d}\lambda,$$

and $\mu_{i,j}^{(k)}$ denote the (i, j)th mixed modified moment for the kth weight function,

$$\mu_{i,j}^{(k)} = \int_\nu p_i{}^*(\lambda) p_j{}^*(\lambda)\omega_k(\lambda)\mathrm{d}\lambda.$$

Substituting Eq. (14) into Eq. (15) and using linearity yields

$$\gamma_i \mu_{l,j}^{(k)} + \gamma_{l+1}\mu_{i+1,j}^{(k)} + \cdots + \gamma_n \mu_{n,j}^{(k)} = 0. \tag{16}$$

Moreover, $\gamma_n \neq 0$ because $q(\lambda)$ has n roots. Without loss of generality, assume that $\gamma_n = 1$. Equation (16) becomes

$$\gamma_l \mu_{l,j}^{(k)} + \gamma_{l+1}\mu_{l+1,j}^{(k)} + \cdots + \gamma_{n-1}\mu_{n-1,j}^{(k)} = -\mu_{n,j}^{(k)}, \tag{17}$$

which gives a system of linear equations that can be solved for the Fourier coefficients γ_i of the quasi-orthogonal polynomial $q(\lambda)$ expanded in terms of the orthogonal polynomial system of $\omega_1(\lambda)$. This method of constructing $q(\lambda)$ is far better numerically than Prony's method. The γ_i must satisfy that

$$\begin{bmatrix}
\mu_{l,0}^{(2)} & \mu_{l+1,0}^{(2)} & \cdots & \mu_{n-1,0}^{(2)} \\
\mu_{l,1}^{(2)} & \mu_{l+1,1}^{(2)} & \cdots & \mu_{n-1,1}^{(2)} \\
\mu_{l,l-1}^{(2)} & \mu_{l+1,l-1}^{(2)} & \cdots & \mu_{n-1,l-1}^{(2)} \\
\mu_{l,0}^{(3)} & \mu_{l+1,0}^{(3)} & \cdots & \mu_{n-1,0}^{(3)} \\
\mu_{l,l-1}^{(3)} & \mu_{l+1,l-1}^{(3)} & \cdots & \mu_{n-1,l-1}^{(3)}
\end{bmatrix}
\begin{bmatrix} \gamma_l \\ \gamma_{l+1} \\ \gamma_{n-1} \end{bmatrix}
=
\begin{bmatrix}
-\mu_{n,0}^{(2)} \\ -\mu_{n,1}^{(2)} \\ -\mu_{n,l-1}^{(2)} \\ -\mu_{n,0}^{(3)} \\ -\mu_{n,l-1}^{(3)}
\end{bmatrix}. \tag{18}$$

I call the matrix in Eq. (18) the mixed modified Gram matrix. Once the elements of this equation have been assembled, the equation can be solved for the Fourier coefficients γ_i.

With the coefficients $\gamma_l, \gamma_{l+1}, \ldots, \gamma_n$ in hand, one can find the roots of the quasi-orthogonal polynomial by solving an eigenvalue problem. Recall that the orthogonal polynomials associated with ω_1 satisfy a three-term recurrence relation of the form

$$x\mathbf{p}(x) = J^{(1)}\mathbf{p}(x) + \beta_n^{(1)} p_n{}^{\natural}(x)\mathbf{e}_n,$$

where $J^{(1)}$ is a Jacobi matrix with elements $\alpha_0^{(1)}, \alpha_1^{(1)}, \ldots, \alpha_{n-1}^{(1)}$ on the diagonal and elements $\beta_1^{(1)}, \beta_2^{(1)}, \ldots,$

$\beta_{n-1}^{(1)}$ on the subdiagonal and where $\mathbf{p}(x) = [p_0^*, p_1^*, \ldots, p_{n-1}^*]^T$ and $\mathbf{e}_n = [0, 0, \ldots, 0, 1]^T$.

Using Eq. (14) and the fact that $\gamma_n = 1$ gives

$$p_n^*(x) = q(x) - \sum_{i=l}^{n-1} \gamma_i p_i^*(x).$$

If we let

$$\mathbf{g} = [0, \ldots, 0, \gamma_l, \ldots, \gamma_{n-1}]^T,$$

then, clearly,

$$x\mathbf{p}(x) = J^{(1)}\mathbf{p}(x) + \beta_n^{(1)}[q(x) - \mathbf{g}^T\mathbf{p}(x)]\mathbf{e}_n,$$

so that the roots x_i of the quasi-orthogonal polynomial are precisely the eigenvalues of the matrix

$$J^{(1)} - \beta_n^{(1)}\mathbf{e}_n\mathbf{g}^T, \tag{19}$$

which can be found stably by use of the QR algorithm.

Notice that the eigenvector associated with x_i is given by $\mathbf{v}_i = \mathbf{p}(x_i)$. This allows us to solve for the weights $w_{k,i}$ by requiring that each rule correctly generate the first n modified moments. This is mathematically equivalent to solving the Vandermonde system for the weights of an interpolatory rule but is much more stable and is convenient, because the eigenvectors are a by-product of the QR algorithm. In particular, let

$$K = [\mathbf{p}(x_1), \mathbf{p}(x_2), \ldots, \mathbf{p}(x_n)]. \tag{20}$$

The vectors $\mathbf{p}(x_i)$ can be produced by scaling the eigenvectors from the QR method so that their first elements are identical and equal to the zero moment of the weight function ω_1. Then one finds the interpolatory weights $\mathbf{w}_k = [w_{k,1}, w_{k,2}, \ldots, w_{k,n}]^T$ by solving

$$K\mathbf{w}_k = \begin{bmatrix} \mu_0^{(k)} \\ \mu_1^{(k)} \\ \vdots \\ \mu_{n-1}^{(k)} \end{bmatrix}. \tag{21}$$

Before presenting the algorithm for constructing these rules, I note that it is possible to compute the modified moments and the mixed modified moments directly from the Jacobi matrices associated with the three weight functions. There is no need to construct the orthogonal polynomials associated with the first weight function or to compute any definite integrals by using them. Golub and Fischer[17] give the following algorithm for computing the modified moments:

Algorithm 3: Let $J^{(k)}$ be the Jacobi matrix of order n for the weight function $\omega_k(\lambda)$. The first $2n$ modified moments

$$\mu_i^{(k)} = \int_\nu p_i^*(\lambda)\omega_k(\lambda)\mathrm{d}\lambda,$$

where $p_i^*(\lambda)$ is the ith element of the orthogonal polynomial sequence associated with $\omega_1(\lambda)$, are given by the following:

$$\text{Set } \mathbf{z}_{-1} = \mathbf{0},\ \mathbf{z}_0 = \mathbf{e}_1,$$
$$\mu_0^{(k)} = \int_\nu \omega_1(\lambda)\mathrm{d}\lambda.$$

For $i = 1, 2 \ldots n$ let

$$\mathbf{z}_i = [J^{(k)} - \alpha_{i-1}^{(1)}I]\mathbf{z}_{i-1} - (\beta_{i-1}^{(k)})^2\mathbf{z}_{i-2},$$
$$\mu_i^{(k)} = \mu_0^{(k)}\mathbf{e}_1{}^T\mathbf{z}_i.$$

After the modified moments are computed, the mixed

Table 2. Gauss Rules for CIE *XYZ* Matching Functions with No Illuminant Bias

	X		*Y*		*Z*	
Order	λ	Weight	λ	Weight	λ	Weight
3	441.1	0.158710	0.487025	0.161582	0.424071	0.319491
	573.6	0.555942	0.559653	0.672840	0.463665	0.628227
	640.9	0.285348	0.633785	0.165577	0.521996	0.052282
4	433.0	0.119848	0.462179	0.046978	0.412796	0.119811
	518.6	0.174684	0.531409	0.487040	0.447986	0.645877
	600.2	0.614486	0.594567	0.427747	0.490538	0.229805
	664.3	0.090982	0.662693	0.038235	0.562460	0.004507
5	422.6	0.061798	0.445422	0.016657	0.399833	0.029414
	465.3	0.120348	0.509732	0.264278	0.434952	0.465719
	572.7	0.453971	0.565290	0.529288	0.469779	0.450593
	628.6	0.350250	0.624277	0.184228	0.516066	0.053619
	695.3	0.013634	0.693947	0.005550	0.592562	0.000655
6	412.7	0.025929	0.432699	0.006338	0.388772	0.007640
	449.7	0.131914	0.489058	0.114729	0.425429	0.272119
	546.2	0.207412	0.541518	0.465343	0.455615	0.537070
	601.2	0.504725	0.594973	0.355536	0.491915	0.172771
	653.0	0.128419	0.650571	0.057452	0.541050	0.010189
	727.3	0.001600	0.726910	0.000602	0.607877	0.000210

Table 3. Shared-Node Rules for the *XYZ* Matching Functions with No Illuminant Bias

Order	λ	Weights X	Y	Z
3	445.4	0.154337	0.035122	0.892225
	540.2	0.257263	0.668486	0.123094
	618.7	0.588400	0.296392	−0.015320
6	424.6	0.060565	0.001520	0.292508
	460.7	0.107723	0.031086	0.629756
	517.5	0.017953	0.302792	0.082737
	567.8	0.367156	0.462950	−0.006311
	618.5	0.399579	0.184717	0.001433
	669.2	0.047024	0.016935	−0.000123
9	412.8	0.014184	0.000401	0.067868
	437.5	0.085663	0.004702	0.427231
	467.6	0.066322	0.027289	0.421487
	505.9	0.001839	0.148927	0.079127
	542.0	0.107973	0.330735	0.003143
	578.2	0.298333	0.295194	0.001145
	613.2	0.315996	0.151496	−0.000015
	649.5	0.100473	0.037923	0.000016
	685.7	0.009218	0.003334	−0.000001

Table 4. Shared-Node Rules for the *XYZ* Matching Functions with No Illuminant Bias

Order	λ	Weights X	Y	Z
12	405.9	0.003545	0.000097	0.016898
	423.4	0.031729	0.001038	0.153443
	442.8	0.070672	0.005311	0.359626
	465.5	0.052995	0.016731	0.325076
	490.4	0.007371	0.053570	0.116537
	518.2	0.013876	0.175890	0.024759
	547.3	0.111094	0.279217	0.003038
	577.5	0.247813	0.250495	0.000481
	607.2	0.284907	0.149095	0.000135
	636.9	0.142553	0.056324	0.000006
	666.7	0.029961	0.010976	0.000000
	692.2	0.003483	0.001257	0.000000

Table 5. Shared-Node Rules for the *XYZ* Matching Functions with No Illuminant Bias

Order	λ	Weights X	Y	Z
15	391.4	0.000013	0.000000	0.000061
	406.6	0.003830	0.000105	0.018258
	422.9	0.028008	0.000900	0.135400
	440.4	0.062538	0.004172	0.315041
	460.7	0.055098	0.012089	0.319006
	481.7	0.016841	0.029311	0.152055
	503.9	0.000342	0.083464	0.046273
	527.2	0.029561	0.185479	0.011581
	552.0	0.110422	0.235785	0.001743
	577.1	0.203962	0.209324	0.000405
	602.0	0.248496	0.141796	0.000158
	627.2	0.168892	0.070770	0.000019
	652.8	0.058476	0.021897	0.000000
	677.1	0.011701	0.004250	0.000000
	695.1	0.001819	0.000656	0.000000

modified moments can be built up by the following recurrence:

$$\mu_{i+1,j}^{(k)} = \mu_{i,j+1}^{(k)} + \{[\alpha_j^{(1)} - \alpha_i^{(1)}]\mu_{i,j}^{(k)} + [\beta_j^{(1)}]^2\mu_{i,j-1}^{(k)} - \beta_j^{(1)}\}^2 \times \mu_{i-1,j}^{(k)}, \tag{22}$$

where

$$\mu_{-1,j}^{(k)} = 0, \qquad \mu_{0,j}^{(k)} = \mu_j^{(k)}, \qquad j = 0, 1, \ldots, 2n.$$

This relation can be verified from the three-term recurrence for the weight function $\omega_1(\lambda)$.

Algorithm 4: Let P_λ be the illuminant bias (with $P_\lambda \equiv 1$ if no illuminant bias is desired).

Table 6. Shared-Node Rules for the *XYZ* Matching Functions with No Illuminant Bias

Order	λ	Weights X	Y	Z
18	401.8	0.000699	0.000019	0.003325
	408.5	0.002870	0.000080	0.013695
	418.8	0.013167	0.000382	0.063340
	431.6	0.039731	0.001743	0.195009
	447.2	0.055318	0.005309	0.287454
	465.1	0.042716	0.012941	0.261283
	484.5	0.012002	0.031483	0.122632
	505.3	0.000429	0.080684	0.039522
	526.0	0.023099	0.157015	0.010674
	547.2	0.079467	0.199378	0.002298
	568.9	0.151325	0.195466	0.000454
	590.7	0.212110	0.153464	0.000242
	612.6	0.198881	0.096310	0.000065
	634.1	0.111990	0.044921	0.000007
	654.8	0.040953	0.015261	0.000000
	673.2	0.011358	0.004139	0.000000
	687.0	0.003115	0.001125	0.000000
	697.6	0.000774	0.000279	0.000000

Table 7. Shared-Node Rules for the *XYZ* Matching Functions with No Illuminant Bias

Order	λ	Weights X	Y	Z
21	401.7	0.000675	0.000019	0.003210
	408.6	0.003124	0.000086	0.014912
	419.1	0.013096	0.000385	0.063017
	430.9	0.034347	0.001453	0.168337
	444.2	0.046671	0.003848	0.238865
	459.1	0.042222	0.008154	0.240366
	474.6	0.022126	0.016571	0.160981
	491.3	0.004291	0.035610	0.069958
	507.6	0.000617	0.063483	0.025332
	522.1	0.011839	0.106794	0.009947
	539.3	0.049471	0.167106	0.003787
	558.9	0.108547	0.187389	0.000793
	578.9	0.167095	0.162975	0.000301
	598.3	0.188323	0.115495	0.000149
	616.4	0.146667	0.067719	0.000036
	631.5	0.074237	0.030302	0.000006
	644.2	0.049103	0.018814	0.000001
	659.8	0.025734	0.009510	0.000000
	675.9	0.008616	0.003134	0.000000
	689.4	0.002548	0.000920	0.000000
	697.9	0.000651	0.000235	0.000000

Table 8. Means and Standard Deviations of CIE $L^*u^*v^*$ Color Differences between Tristimulus Values Computed with 1-nm Riemann Summation and Shared-Node Rules[a]

	Nickerson		Macbeth	
Order	μ	σ	μ	σ
3	6.0421	4.3969	7.5159	6.3821
6	2.0323	1.7020	1.8732	1.6587
9	0.6693	0.7503	0.5741	0.5146
12	0.2937	0.2782	0.3417	0.3404
15	0.1568	0.1033	0.1939	0.1345
18	0.0819	0.0571	0.1599	0.0905
21	0.0615	0.0429	0.1183	0.0825

[a]Data are for 462 spectral reflectances of Munsell chips collected by Nickerson and for 24 spectral reflectances from the Macbeth Color Checker: μ, mean; σ, standard deviation.

1. Use algorithm 3 to generate the modified moments of $P_\lambda\omega_2(\lambda)$ and $P_\lambda\omega_3(\lambda)$ with respect to $P_\lambda\omega_1(\lambda)$ as high as order $2n$.
2. Use the relation in Eq. (22) to generate the mixed modified moments of $P_\lambda\omega_2(\lambda)$ and $P_\lambda\omega_3(\lambda)$.
3. Assemble Eq. (18), using the mixed modified moments generated in the step 2, and solve for the Fourier coefficients $\gamma_1, \ldots, \gamma_{n-1}$ of the quasi-orthogonal polynomial.
4. Use the QR algorithm to find the eigenvalues and eigenvectors of

$$J^{(1)} - \beta_n^{(1)}\mathbf{e}_n\mathbf{g}^T,$$

where

$$\mathbf{g} = [0, \ldots, 0, \gamma_1, \ldots, \gamma_{n-1}]^T.$$

The eigenvalues are the shared nodes.
5. Substitute the eigenvectors from step 4 and the modified moments computed above into Eq. (21), and solve for the weights of the three rules. Scale the weights for $P_\lambda\omega_k(\lambda)$ by

$$\int_\nu P_\lambda\omega_k(\lambda)\mathrm{d}\lambda$$

for $k = 1, 2, 3$.

There are conditions on the weight functions that guarantee that the shared nodes will be real, distinct, and in the interval of integration. The details can be found in Ref. 18 and are somewhat tedious; however, practical experience shows that these rules generally exist when 30 or fewer points are desired. The rules of orders 3, 6, 9, 12, 15, 18, 21, and 24 for the XYZ matching functions appear in Tables 1–8. If more than 30 points are desired, then it is best to partition the visible interval and construct a composite rule. In this way, more-accurate rules can be found.

As a final note I point out that experience has shown that it is best to forgo step 6 from algorithm 4 and to use algorithm 2 to calculate the weights. This allows one to avoid accruing too many numerical errors in the calculation of the weights. In particular, the eigenvectors generated in algorithm 4 may be somewhat inaccurate (unless some inverse iteration is used to clean them up), and these inaccuracies will be introduced into the weights. Therefore it is better to start from scratch during the computation of the weights. The weights that appear in the tables were computed in this way.

6. PROPERTIES OF THE SHARED-NODE RULES

In addition to having the highest overall degree of precision, the shared-node rules have another interesting property. In particular, the nodes are invariant under any linear change of color primaries. Moreover, the weights for the new rules can be easily calculated from those of the original rule.

For example, the transformation from XYZ to red–green–blue (RGB) is given mathematically by[19]

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.9107 & -0.5326 & -0.2883 \\ -0.9843 & 1.9984 & -0.0283 \\ 0.0583 & -0.1185 & 0.8986 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}.$$

It is quite simple to verify that the shared nodes are invariant by noting that each new primary can be represented as a linear combination of the old primaries. Hence the new matching functions $\hat{\omega}_k(\lambda)$ are given by

$$\hat{\omega}_k(\lambda) = c_{k,1}\omega_1(\lambda) + c_{k,2}\omega_2(\lambda) + c_{k,3}\omega_3(\lambda),$$

where the $c_{k,i}$ are scalars.

Also, if $q(\lambda)$ is a polynomial of exact degree $3n$ whose roots are the shared nodes for a rule of degree $3n$ in the original primaries, then

$$\int_\nu p(\lambda)q(\lambda)\omega_k(\lambda)\mathrm{d}\lambda = 0$$

for all polynomials $p(\lambda)$ of degree less than n. Finally, note that

$$\begin{aligned}\int_\nu p(\lambda)q(\lambda)\hat{\omega}_k(\lambda)\mathrm{d}\lambda &= c_{k,1}\int_\nu p(\lambda)q(\lambda)\omega_1(\lambda)\mathrm{d}\lambda \\ &\quad + c_{k,2}\int_\nu p(\lambda)q(\lambda)\omega_2(\lambda)\mathrm{d}\lambda \\ &\quad + c_{k,3}\int_\nu p(\lambda)q(\lambda)\omega_3(\lambda)\mathrm{d}\lambda \\ &= 0;\end{aligned}$$

hence the roots of $q(\lambda)$ are also the shared nodes for a rule of degree $3n$ in the new primaries. The weights for the shared-node rules transformed into a new color space can be reduced from the originals by application of the same linear transformation.

7. EXPERIMENTS WITH SHARED-NODE RULES

A number of numerical experiments were conducted with the shared-node rules described in Sections 5 and 6. The test data consist of the spectral reflectances of 462 Munsell chips collected by Nickerson and the 24 spectral reflectances of the Macbeth Color Checker. Table 8 shows the means and standard deviations of the CIE $L^*u^*v^*$ color differences between tristimulus values computed with shared-node rules and the exact values computed by Riemann summation at 1 nm intervals. Note that all experiments were performed with an assumed spectral bandpass of 1 nm. The chromaticity diagrams in Figs. 1–3 show small line segments joining the chromaticity coordinates of the approximate tristimulus values to those of the exact coordinates for the elements of the Macbeth Color Checker. Only 3-, 6-, and 9-point rules are shown, because beyond that the shifts are too small to plot. The results for the Nickerson data are essentially identical.

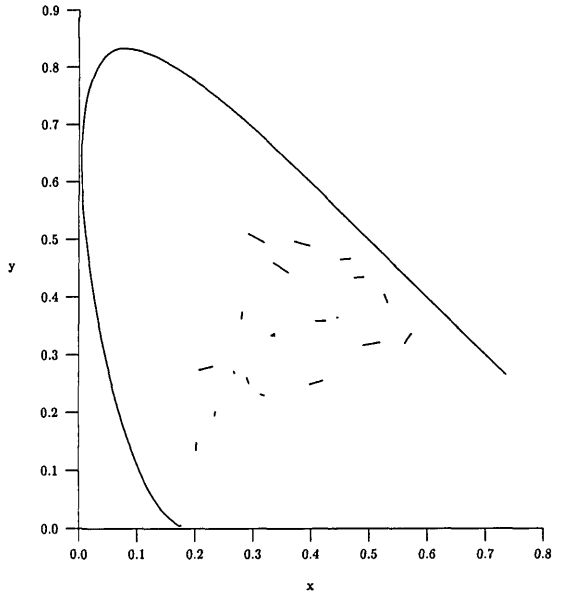

Fig. 1. Plot of the chromaticity shifts for the elements of the Macbeth Color Checker with use of the three-point shared-node rule.

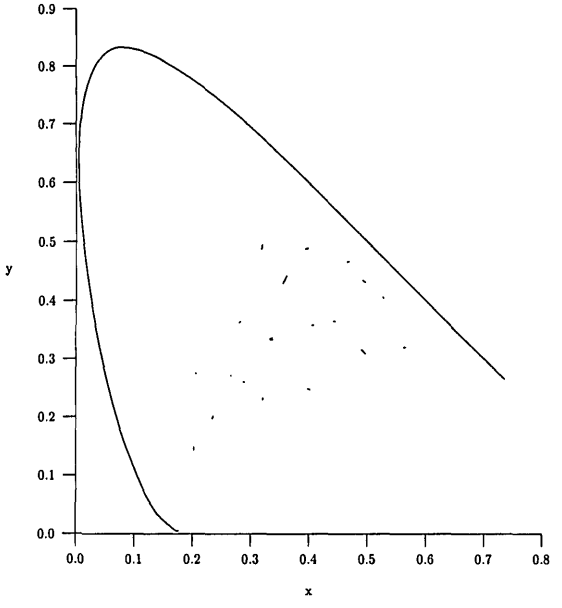

Fig. 2. Plot of the chromaticity shifts for the elements of the Macbeth Color Checker with use of the six-point shared-node rule.

8. OVERVIEW AND CONCLUSIONS

I have examined the problem of constructing approximate quadrature rules for the computation of tristimulus values. Of the three methods considered, the last is, to my knowledge, entirely new and was developed specifically for the solution of this problem. The best available numerical algorithms for constructing all three types of rule were given. All three rely on the use of Jacobi matrices, which are known to have much better numerical properties than moment matrices. All the methods that were outlined allow for the use of arbitrary illuminant biases, which adds to their utility. I introduced the notion of a performance ratio and showed that the shared-node rules maximize this ratio in a well-defined sense. Last, I demonstrated an interesting invariance property of the shared-node rules.

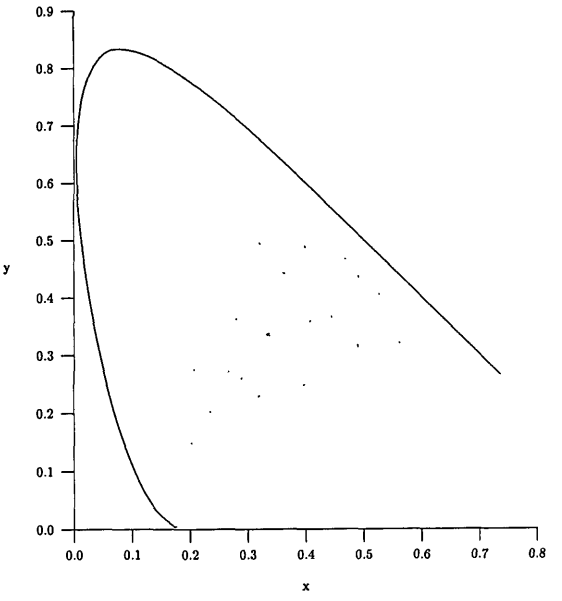

Fig. 3. Plot of the chromaticity shifts for the elements of the Macbeth Color Checker with use of the nine-point shared-node rule.

All the algorithms discussed above have been implemented and extensively tested in the C programing language and with the Matlab package.

ACKNOWLEDGMENTS

The author was supported by a direct grant from the Naval Postgraduate School and acknowledges support from the Computer Graphics Research Laboratory at the University of California, Davis. The author is grateful to both Mark Peercy and Brian Wandell of Stanford University for providing the Nickerson and Macbeth reflectance data used in this study.

REFERENCES

1. G. Wyszecki and W. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae,* 2nd ed. (Wiley, New York, 1982).
2. P. Moon and D. Spencer, "Polynomial representation of reflectance curves," J. Opt. Soc. Am. **35,** 597–600 (1945).
3. R. Wallis, "Fast computation of tristimulus values by use of Gaussian quadrature," J. Opt. Soc. Am. **65,** 91–94 (1975).
4. E. Stearns, "The influence of spectral bandpass on accuracy of tristimulus data," Color Res. Appl. **12,** 282–284 (1987).
5. W. Venable, "Accurate tristimulus values from spectral data," Color Res. Appl. **14,** 260–267 (1989).
6. P. Davis and P. Rabinowitz, *Methods of Numerical Integration,* 2nd ed. (Academic, New York, 1983).
7. G. Golub and J. Welsch, "Calculation of Gauss quadrature rules," Math. Comp. **23,** 221–230 (1969).
8. H. Wilf, *Mathematics for the Physical Sciences* (Wiley, New York, 1962).
9. J. Francis, "The $q-r$ transformation: a unitary analogue to the $l-r$ transformation. i, ii," Comput. J. **4,** 265–271 (1961/1962).
10. W. Gautschi, "On the construction of Gaussian quadrature rules from modified moments," Math. Computa. **24,** 245–260 (1970).

11. W. Gautschi, "Construction of Gauss–Christoffel quadrature formulas," Math. Comp. **22,** 251–270 (1968).
12. W. Gragg and W. Harrod, "The numerically stable reconstruction of Jacobi matrices from spectral data," Numer. Math. **44,** 317–335 (1984).
13. G. Golub and C. V. Loan, *Matrix Computations,* 2nd ed. (Cambridge U. Press, Cambridge, 1990).
14. G. Galimberti and V. Pereyra, "Solving confluent Vandermonde systems of Hermite type," Numer. Math. **18,** 44–60 (1971).
15. J. Kautsky and S. Elhay, "Calculation of the weights of interpolatory quadratures," Numer. Math. **40,** 407–422 (1982).
16. P. Davis, *Interpolation and Approximation* (Dover, New York, 1975).
17. G. Golub and B. Fischer, "On generating polynomials which are orthogonal over several intervals," Math. Computa. **56,** 711–730 (1991).
18. C. Borges, "On a class of Gauss-like quadrature rules," Numer. Math. **67,** 271–288 (1994).
19. G. Wyszecki and D. Judd, *Color in Business, Science, and Industry* (Wiley, New York, 1975).